DESIGN AND SIMULATION OF AN ALARM CLOCK

Name: Soham Sen
Scholar Id:2224211

Department of Electronics and Communication Engineering
National Institute of Technology Silchar

AIM: Design and Simulation of an Alarm Clock

SOFTWARE USED: XILINX VIVADO 2022.1

THEORY: Verilog code for an alarm clock is presented in this project. The simple alarm clock black box is shown in the following figure. The alarm clock output is a real-time clock with a 24-hour format and provides an alarm feature. Users also can set the clock time through switches.

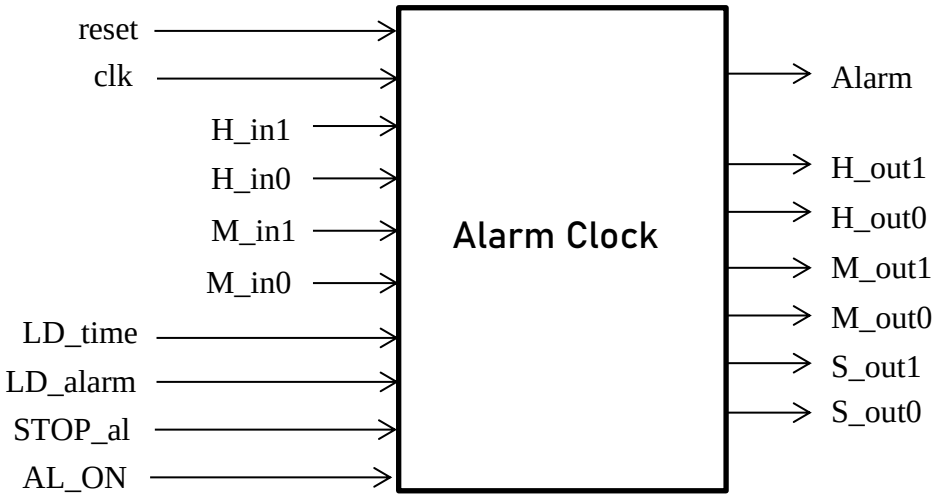


Fig. 1. Model of Alarm Clock

Active high '**reset**' pulse, to set the time to the input hour and minute (as defined by the '**H_in1'**, '**H_in0'**, '**M_in1**', and '**M_in0**' inputs) and the second to 00. It should also set the alarm value to 0.00.00, and to set the '**Alarm'** (output) low.For normal operation, this input pin should be 0.

A 10Hz input '**clk**' signal is clock. This should be used to generate each real-time second.

A 2-bit input '**H_in1'** signal is used to set the most significant hour digit of the clock (if '**LD_time**'=1),or the most significant hour digit of the alarm (if '**LD_alarm**'=1). Valid values are 0 to 2.

A 4-bit input '**H_in0**' signal is used to set the least significant hour digit of the clock (if '**LD_time'**=1),or the least significant hour digit of the alarm (if '**LD_alarm**'=1). Valid values are 0 to 9.

A 4-bit input '**M_in1**' signal is used to set the most significant minute digit of the clock (if '**LD_time'**=1),or the most significant minute digit of the alarm (if '**LD_alarm**'=1). Valid values are 0 to 5.

A 4-bit input '**M_in0**' signal is used to set the least significant minute digit of the clock (if '**LD_time'**=1),or the least significant minute digit of the alarm (if '**LD_alarm**'=1). Valid values are 0 to 9.

If '**LD_time=1'**, the time should be set to the values on the inputs '**H_in1'**, '**H_in0'**, '**M_in1'**, and '**M_in0'**. The second time should be set to 0.If '**LD_time'**=0, the clock should act normally (i.e. second should be incremented every 10 clock cycles).

If '**LD_alarm'**=1, the alarm time should be set to the values on the inputs '**H_in1'**, '**H_in0'**, '**M_in1'**, and '**M_in0'**.If '**LD_alarm'**=0, the clock should act normally.

If the '**Alarm'** (output) is high, then '**STOP_al'**=1 will bring the output back low.

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

This 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.

'H_out1' signal is used for the most significant digit of the hour. Valid values are 0 to 2.

'H_out0' signal is used for the least significant digit of the hour. Valid values are 0 to 9.

‘M_out1’ signal is for the most significant digit of the minute. Valid values are 0 to 5.

‘M_out0’ signal is for the least significant digit of the minute. Valid values are 0 to 9.

‘**S_out1’** signal is for the most significant digit of the minute. Valid values are 0 to 5.

‘S_out0’ signal is for the least significant digit of the minute. Valid values are 0 to 9.

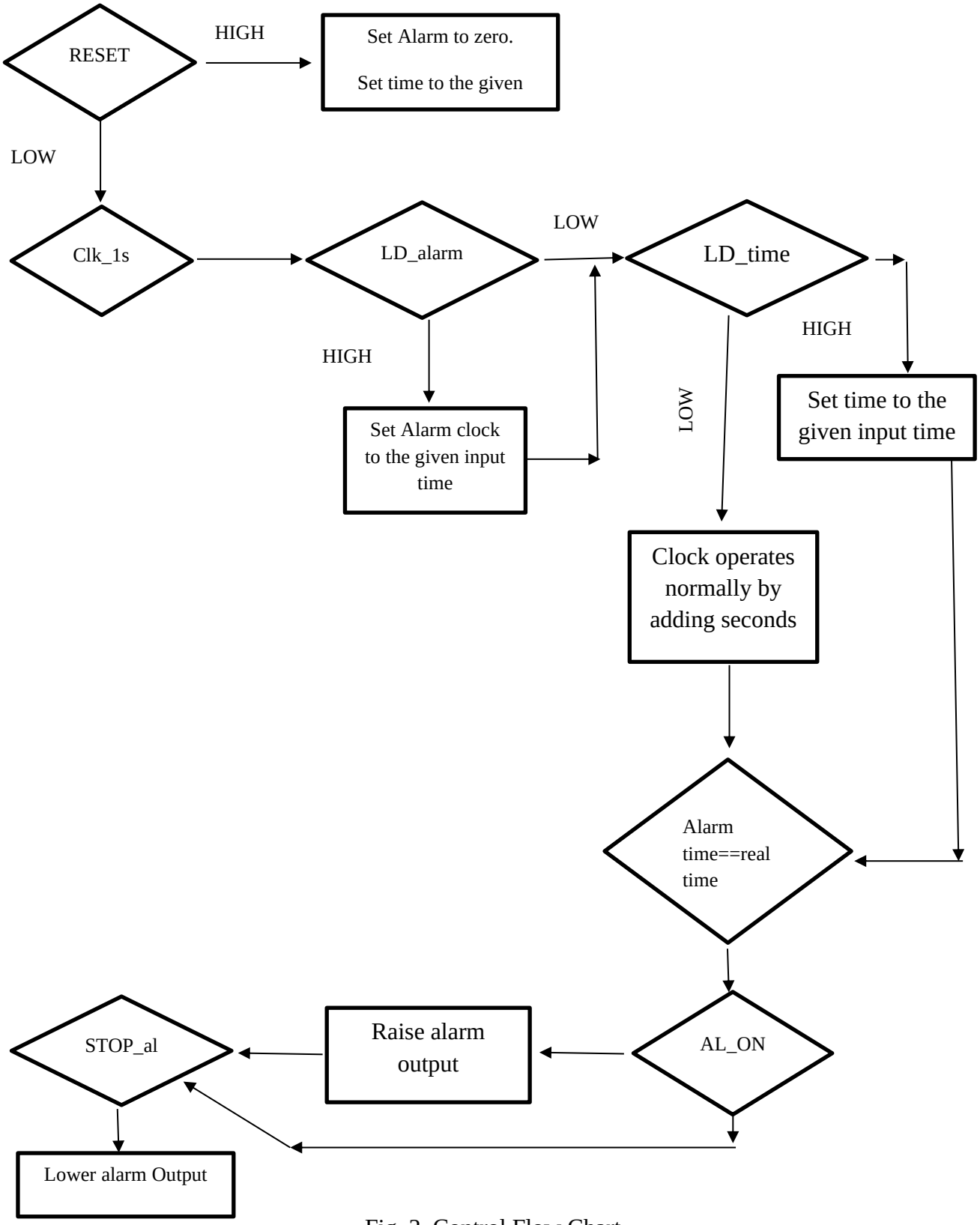


Fig. 2. Control Flow Chart

VERILOG CODE:

Design

```
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
```

```
//Loading and incrementing time
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

//Make 1s clock
 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


//Setting and disabling alarm
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

//Output time
 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

```
module aclock_tb;


 // Inputs
 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;


// Instantiate the Unit Under Test (UUT)

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)
        );

// clock 10Hz
initial
begin
        clk = 0;
        forever #50000000 clk = ~clk;
end
initial
begin

// Initialize Inputs
        reset = 1;
        H_in1 = 0;
        H_in0 = 2;
        M_in1 = 0;
        M_in0 = 4;
```

```
 LD_time = 0;
LD_alarm = 0;
STOP_al = 0;
AL_ON = 0; // set clock time to 02h04, alarm time to 00h00 when reset
                    // Wait 100 ns for global reset to finish
#1000000000
 reset = 0;
 H_in1 = 0;
H_in0 = 2;
 M_in1 = 0;
 M_in0 = 5;
 LD_time = 0;
 LD_alarm = 1;
 STOP_al = 0;
 AL_ON = 1; // turn on Alarm and set the alarm time to 02h05
 wait(Alarm); // wait until Alarm signal is high when the alarm time equals clock time
#1000000000
 STOP_al = 1; // pulse high the STOP_al to push low the Alarm signal
 end
endmodule
```

RTL VIEW:

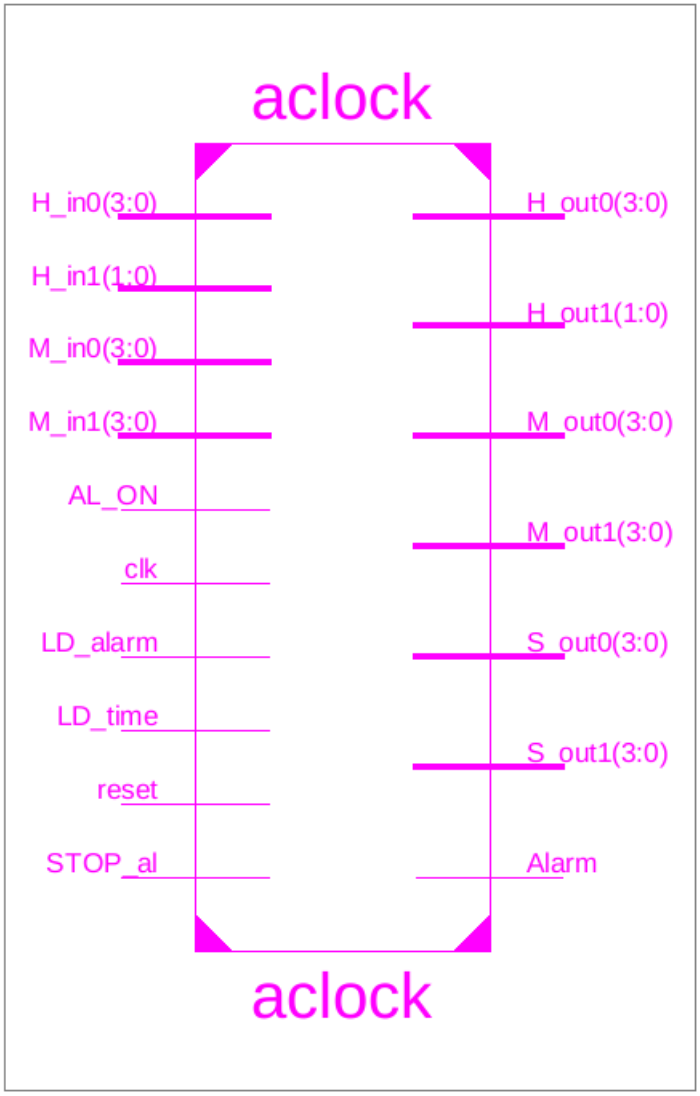


Fig. 3. Block Diagram

INTERNAL RTL VIEW:

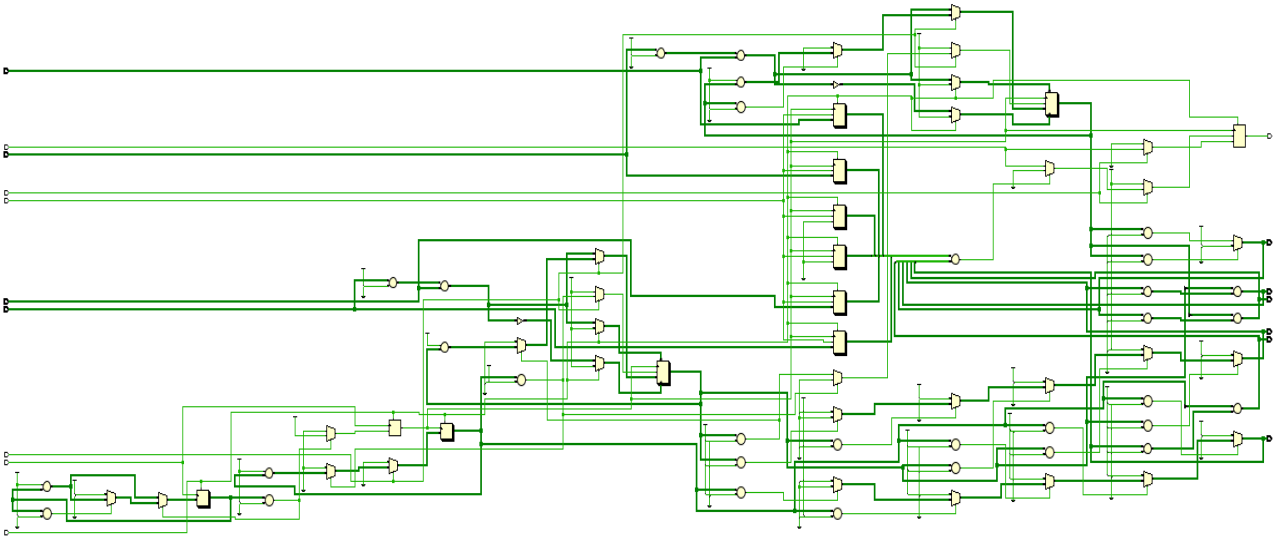
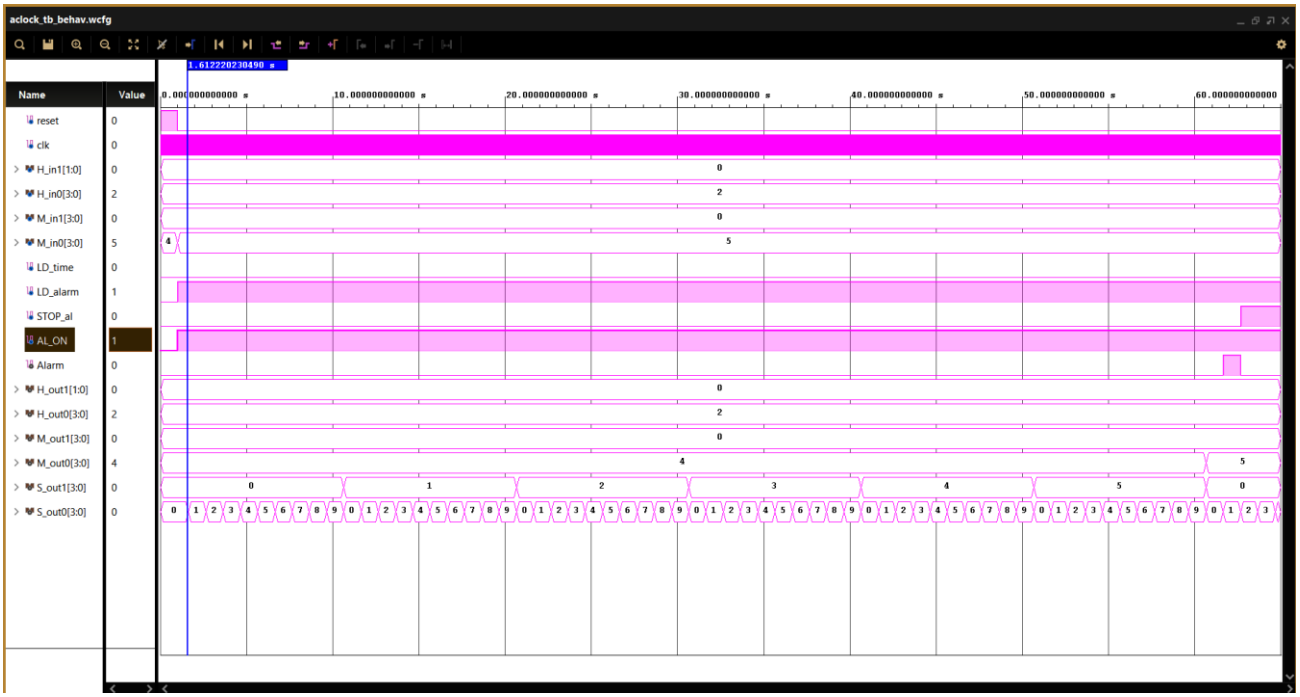
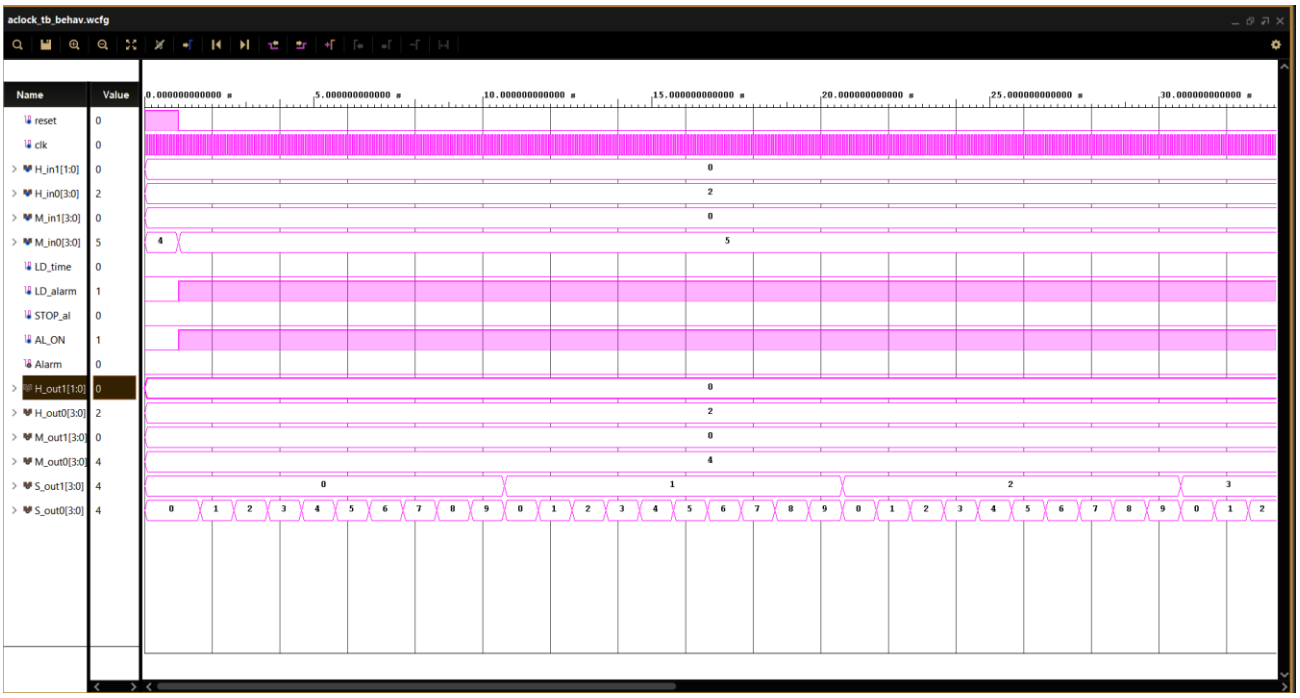


Fig. 4. Elaborated Design Schematic

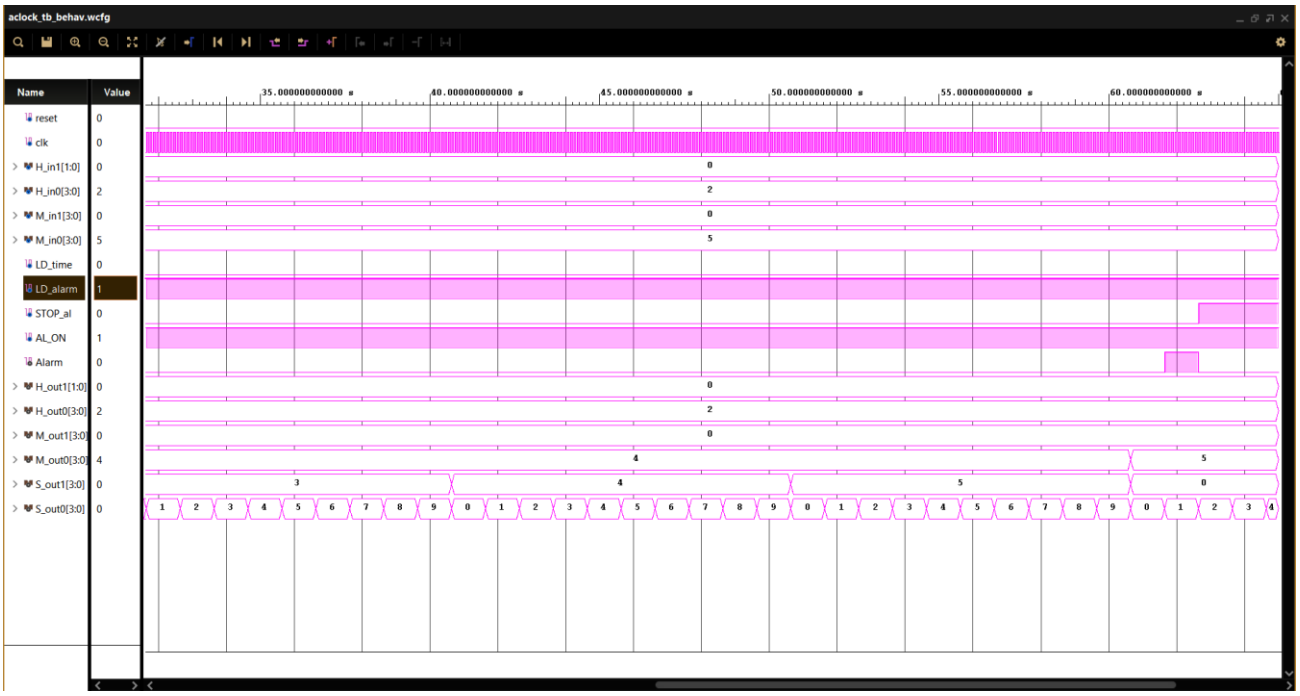
SIMULATION WAVEFORMS:



(a)



(b)



(c)

Fig. 5. Simulation waveform

SYNTHESIS REPORT:

Project Device: xc7a100tcsg324-3

Product Family: Artix-7

Device Utilization Summary:

Utilization — Post-Synthesis | Post-Implementation

Graph | Table

Resource	Estimation	Available	Utilization %
LUT	112	63400	0.18
FF	50	126800	0.04
IO	43	210	20.48
BUFG	2	32	6.25

Fig. 6. Device Utilization Table

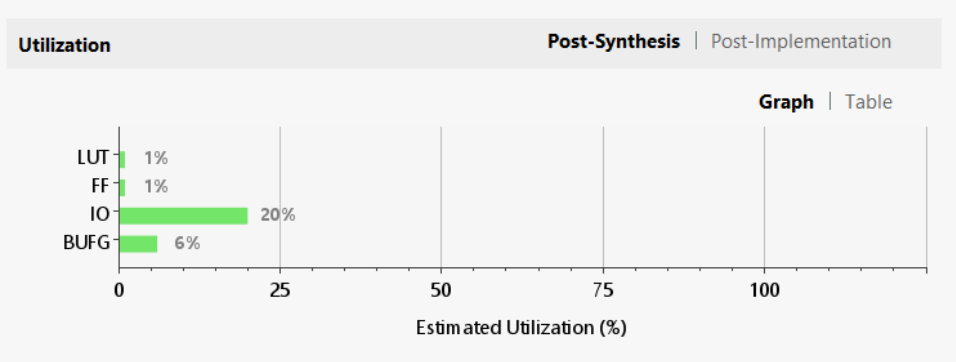


Fig. 7. Device Utilization Graph

Timing Summary:

Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	inf	Worst Hold Slack (WHS):	inf	Worst Pulse Width Slack (WPWS):	NA
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	NA
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	NA
Total Number of Endpoints:	160	Total Number of Endpoints:	160	Total Number of Endpoints:	NA

There are no user specified timing constraints.

Fig. 8. Device Timing Summary

Power:

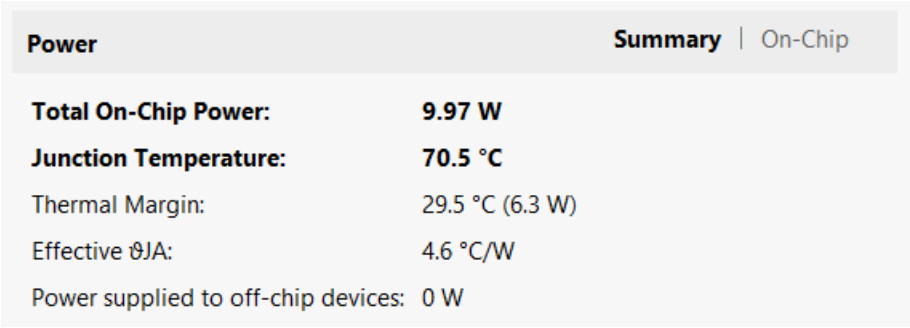


Fig. 9. Power Summary

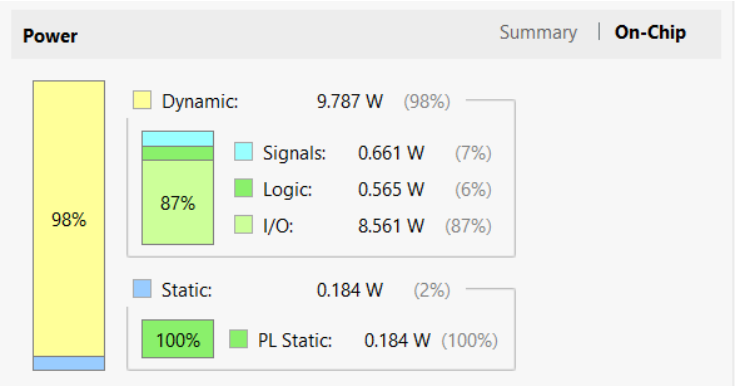


Fig. 10. On-Chip Power

REFERENCES:

[1] Samir Palnitkar. *Verilog HDL: A Guide to Digital Design and Synthesis*. Prentice Hall, New York, 1996.

[2] Revati Muley , Bhushan Patil and Rabinder Henry, “Design and Implementation of Digital Clock with Stopwatch on FPGA”, IEEE International Conference on Intelligent Computing and Control Systems 2017.