DIGITAL & VLSI LAB
PROJECT REPORT

Topic **Digital Alarm Clock**

A team effort by,

AKSHAT SRIVASTAVA (05)

TUSHAR KANT (47)

VIKAS KUMAR SAHA (62)

PROJECT OVERVIEW

Under this project, we have designed an alarm clock which sets alarm time by taking input from the user in binary. The alarm clock has been implemented digitally using RTL code in Vivado software, and further tested on the PYNQ board. Due to input port constraints in the PYNQ board, we had to restrict our time implementation upto an alarm time of 15 seconds or a 4 – bit alarm time input.

The user needs to provide input in binary form. For example, an alarm time of 11 seconds should be input given in the binary form of 1011 in an appropriate way using the input switches as specified. Once the alarm is set, using similar mechanism the alarm clock is set on and the timer begins counting seconds and as soon counting exceeds 11, the specified alarm clock signal changes colour signifying alarm time reached. This output can also be externally connected to a buzzer system, making this digital clock more practical.

Overall, the project yields desired purposes. The input time limit can be exceeded from 15 seconds to minutes and even hours, provided sufficient input ports are available. Buzzer system can help the clock yield more effectiveness.

AIM

To design and implement a digital clock using Verilog on PYNQ board.

RTL CODE

```
`timescale 1ns / 1ps


module new_clk(Oclk,Iclk);          // 1Hz
output reg Oclk;
input Iclk;
integer counter;
initial counter = 0;
initial Oclk = 0;
always @(posedge Iclk)
begin
    if (counter < 125000000) counter <= counter + 1;
    else counter = 0;
    Oclk <= (counter < 62500000) ? 1 : 0;
end
endmodule

module start_alarm_clk(Ring, sec3, sec2, sec1, sec0, Clk, Confirmation);     //start alarm
output reg Ring;
input Clk, Confirmation, sec3, sec2, sec1, sec0;
integer counter; initial counter = 0;
always @(posedge Clk)
begin
    if (Confirmation)
    begin
        if (sec3 == 0 && sec2 == 0 && sec1 == 0 &&  sec0 == 0)
        begin
            if (counter < 0) begin counter <= counter + 1; Ring <= 0; end
            else Ring <= 1;
        end
        else if (sec3 == 0 && sec2 == 0 && sec1 == 0 &&  sec0 == 1)
        begin
            if (counter < 1) begin counter <= counter + 1; Ring <= 0; end
            else Ring <= 1;
        end
        else if (sec3 == 0 && sec2 == 0 && sec1 == 1 &&  sec0 == 0)
        begin
            if (counter < 2) begin counter <= counter + 1; Ring <= 0; end
            else Ring <= 1;
        end
        else if (sec3 == 0 && sec2 == 0 && sec1 == 1 &&  sec0 == 1)
```

```
41          else if (sec3 == 0 && sec2 == 0 && sec1 == 1 &&  sec0 == 1)
42          begin
43              if (counter < 3) begin counter <= counter + 1; Ring <= 0; end
44              else Ring <= 1;
45          end
46          else if (sec3 == 0 && sec2 == 1 && sec1 == 0 &&  sec0 == 0)
47          begin
48              if (counter < 4) begin counter <= counter + 1; Ring <= 0; end
49              else Ring <= 1;
50          end
51          else if (sec3 == 0 && sec2 == 1 && sec1 == 0 &&  sec0 == 1)
52          begin
53              if (counter < 5) begin counter <= counter + 1; Ring <= 0; end
54              else Ring <= 1;
55          end
56          else if (sec3 == 0 && sec2 == 1 && sec1 == 1 &&  sec0 == 0)
57          begin
58              if (counter < 6) begin counter <= counter + 1; Ring <= 0; end
59              else Ring <= 1;
60          end
61          else if (sec3 == 0 && sec2 == 1 && sec1 == 1 &&  sec0 == 1)
62          begin
63              if (counter < 7) begin counter <= counter + 1; Ring <= 0; end
64              else Ring <= 1;
65          end
66          else if (sec3 == 1 && sec2 == 0 && sec1 == 0 &&  sec0 == 0)
67          begin
68              if (counter < 8) begin counter <= counter + 1; Ring <= 0; end
69              else Ring <= 1;
70          end
71          else if (sec3 == 1 && sec2 == 0 && sec1 == 0 &&  sec0 == 1)
72          begin
73              if (counter < 9) begin counter <= counter + 1; Ring <= 0; end
74              else Ring <= 1;
75          end
76          else if (sec3 == 1 && sec2 == 0 && sec1 == 1 &&  sec0 == 0)
77          begin
78              if (counter < 10) begin counter <= counter + 1; Ring <= 0; end
79              else Ring <= 1;
80          end
81          else if (sec3 == 1 && sec2 == 0 && sec1 == 1 &&  sec0 == 1)
```

```
        end
        else if (sec3 == 1 && sec2 == 0 && sec1 == 1 &&  sec0 == 1)
        begin
            if (counter < 11) begin counter <= counter + 1; Ring <= 0; end
            else Ring <= 1;
        end
        else if (sec3 == 1 && sec2 == 1 && sec1 == 0 &&  sec0 == 0)
        begin
            if (counter < 12) begin counter <= counter + 1; Ring <= 0; end
            else Ring <= 1;
        end
        else if (sec3 == 1 && sec2 == 1 && sec1 == 0 &&  sec0 == 1)
        begin
            if (counter < 13) begin counter <= counter + 1; Ring <= 0; end
            else Ring <= 1;
        end
        else if (sec3 == 1 && sec2 == 1 && sec1 == 1 &&  sec0 == 0)
        begin
            if (counter < 14) begin counter <= counter + 1; Ring <= 0; end
            else Ring <= 1;
        end
        else if (sec3 == 1 && sec2 == 1 && sec1 == 1 &&  sec0 == 1)
        begin
            if (counter < 15) begin counter <= counter + 1; Ring <= 0; end
            else Ring <= 1;
        end
    end
    else begin Ring <= 0; counter <= 0; end
end
endmodule

module takedata(Seconds, signal, Input, Manual_cnf);                // get time in seconds
output reg [3:0]Seconds;
output reg signal;
input Manual_cnf, Input;
integer counter; initial counter = 0;
always@(posedge Manual_cnf)
begin
    if (counter < 5) counter <= counter + 1 ;
    else begin counter <= 0; signal <= 0; end
    if (counter == 0) begin Seconds[0] <= Input; signal <= 0; end
```

```
97          begin
98              if (counter < 14) begin counter <= counter + 1; Ring <= 0; end
99              else Ring <= 1;
100         end
101         else if (sec3 == 1 && sec2 == 1 && sec1 == 1 &&  sec0 == 1)
102         begin
103             if (counter < 15) begin counter <= counter + 1; Ring <= 0; end
104             else Ring <= 1;
105         end
106     end
107     else begin Ring <= 0; counter <= 0; end
108 end
109 endmodule
110
111 module takedata(Seconds, signal, Input, Manual_cnf);                  // get time in seconds
112 output reg [3:0]Seconds;
113 output reg signal;
114 input Manual_cnf, Input;
115 integer counter; initial counter = 0;
116 always@(posedge Manual_cnf)
117 begin
118     if (counter < 5) counter <= counter + 1 ;
119     else begin counter <= 0; signal <= 0; end
120     if (counter == 0) begin Seconds[0] <= Input; signal <= 0; end
121     else if (counter == 1) begin Seconds[1] <= Input; signal <= 0; end
122     else if (counter == 2) begin Seconds[2] <= Input; signal <= 0; end
123     else if (counter == 3) begin Seconds[3] <= Input; signal <= 0; end
124     else if (counter == 4) signal <= 1;
125 end
126 endmodule
127
128
129 module test(on, clk1, start, time_s, in, manual_clk, clk);
130 output on, clk1, start;
131 output [3:0]time_s;
132 input in;
133 input manual_clk, clk;
134 new_clk Hz1(clk1,clk);
135 takedata TD(time_s, start, in, manual_clk);
136 start_alarm_clk Alarm(on, time_s[3], time_s[2], time_s[1], time_s[0], clk1, start) ;
137 endmodule
```

BRIEF CODE EXPOSURE

The module new_clk generates an output clock signal (Oclk) based on an input clock signal (Iclk). It uses a counter that increments until a threshold is reached, at which point it resets back to zero. The Oclk output toggles between high and low based on the counter value.

The module start_alarm clk handles the alarm functionality. The Ring output is activated based on certain conditions and input signals (Confirmation, sec3, sec2, secl, secO).

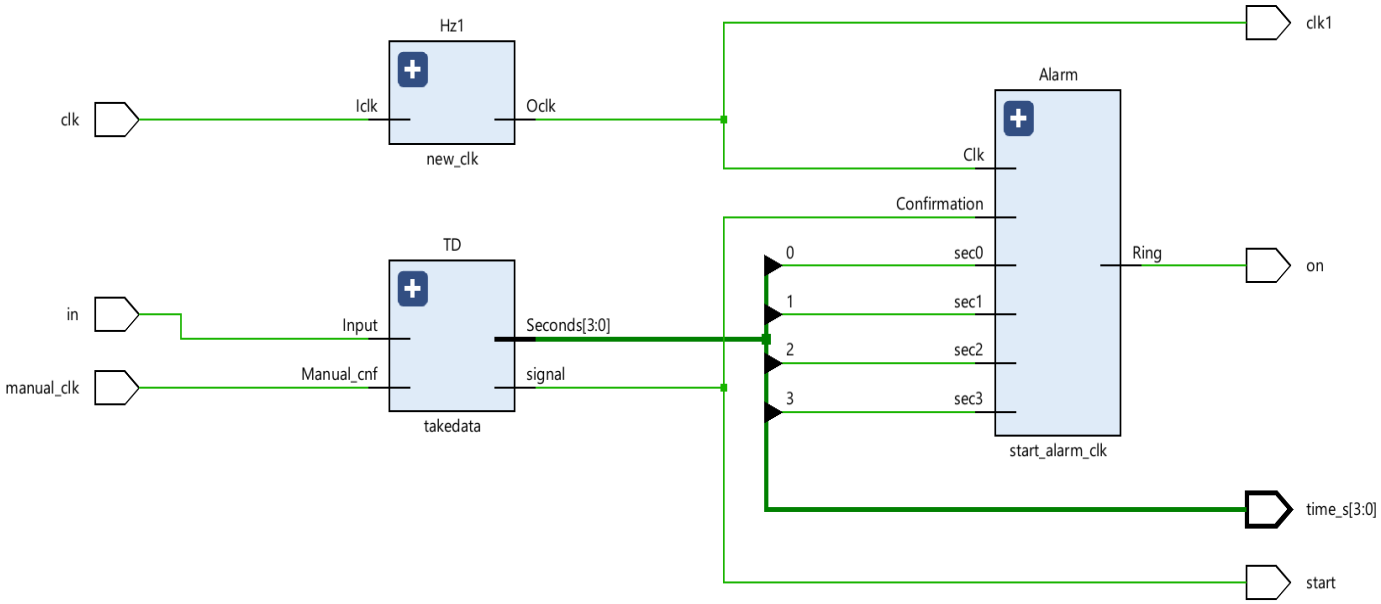
The module takedata is used for input data acquisition, potentially for clock or alarm settings. It updates the Seconds and signal outputs based on input conditions and a counter.

The module test is the top-level module that connects and instantiates other modules (new_clk, takedata, start_alarm clk) to create a clock and alarm system.

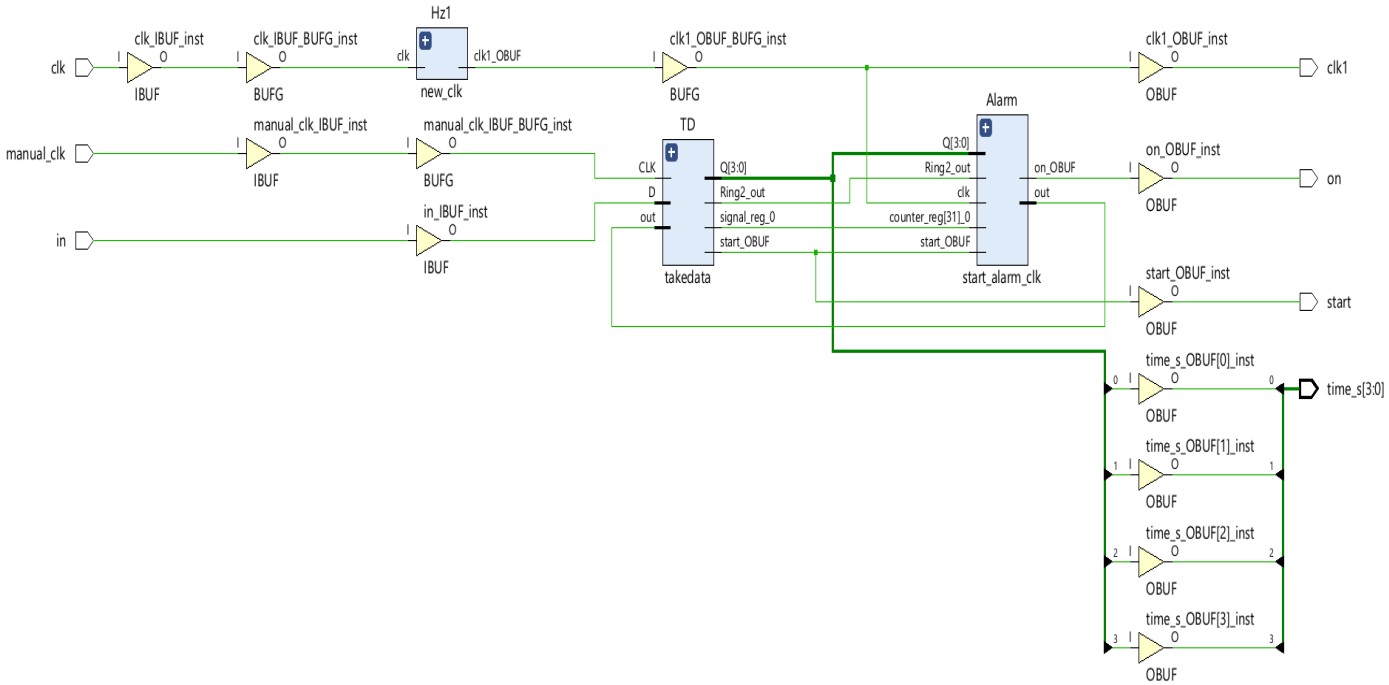
TESTBENCH

```
11  // Tool Versions:
12  // Description:
13  //
14  // Dependencies:
15  //
16  // Revision:
17  // Revision 0.01 - File Created
18  // Additional Comments:
19  //
20  //////////////////////////////////////////////////////////////////////////////////
21
22
23  module tb_project();
24      wire on, clk1, start;
25      wire [3:0]time_s;
26      reg in, clk, manual_clk;
27      test dut(on, clk1, start, time_s, in, manual_clk, clk);
28
29  initial
30      begin
31          clk = 1;
32          in = 0;
33          manual_clk = 0;
34      end
35  always #4 clk = ~clk;
36  initial
37      begin
38          #10 manual_clk = 1; in = 0;
39          #10 manual_clk = 0;
40          #10 manual_clk = 1; in = 1;
41          #10 manual_clk = 0;
42          #10 manual_clk = 1; in = 0;
43          #10 manual_clk = 0;
44          #10 manual_clk = 1; in = 0;
45          #10 manual_clk = 0;
46          #10 manual_clk = 1;
47          #10 manual_clk = 0;
48      end
49  endmodule
50
```

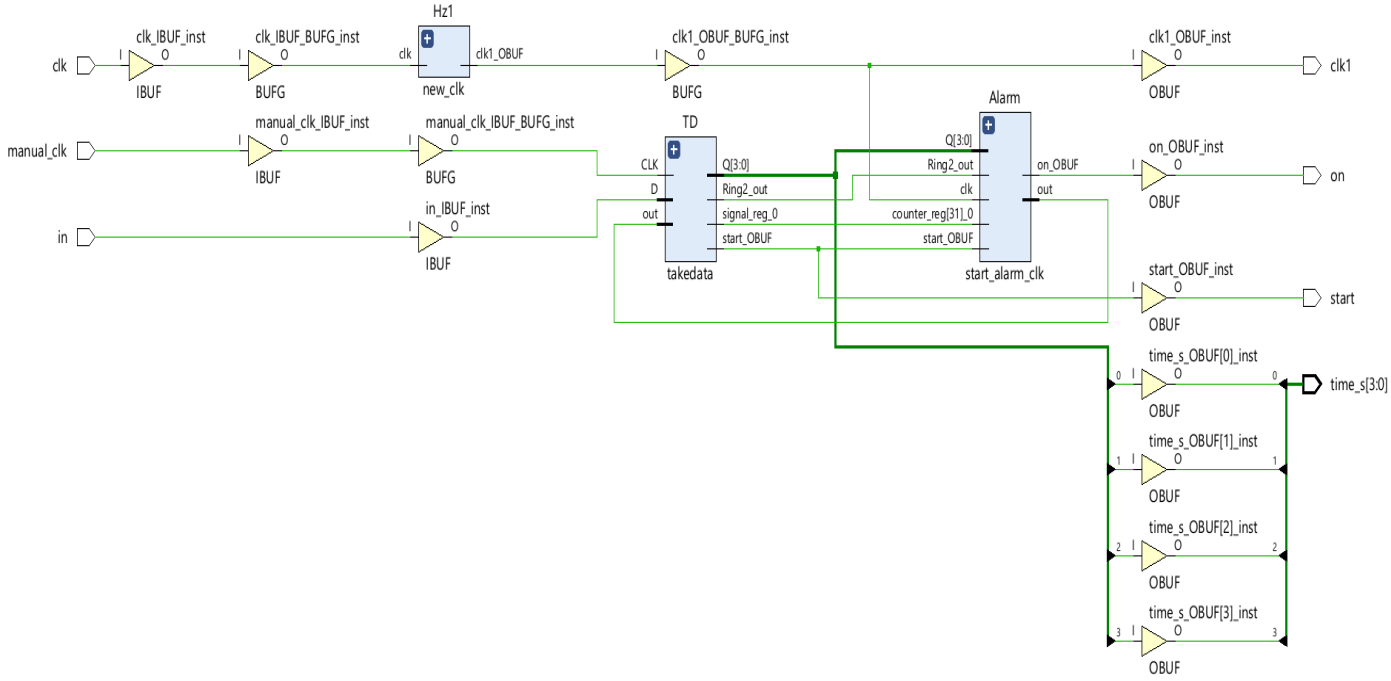
RTL SCHEMATIC



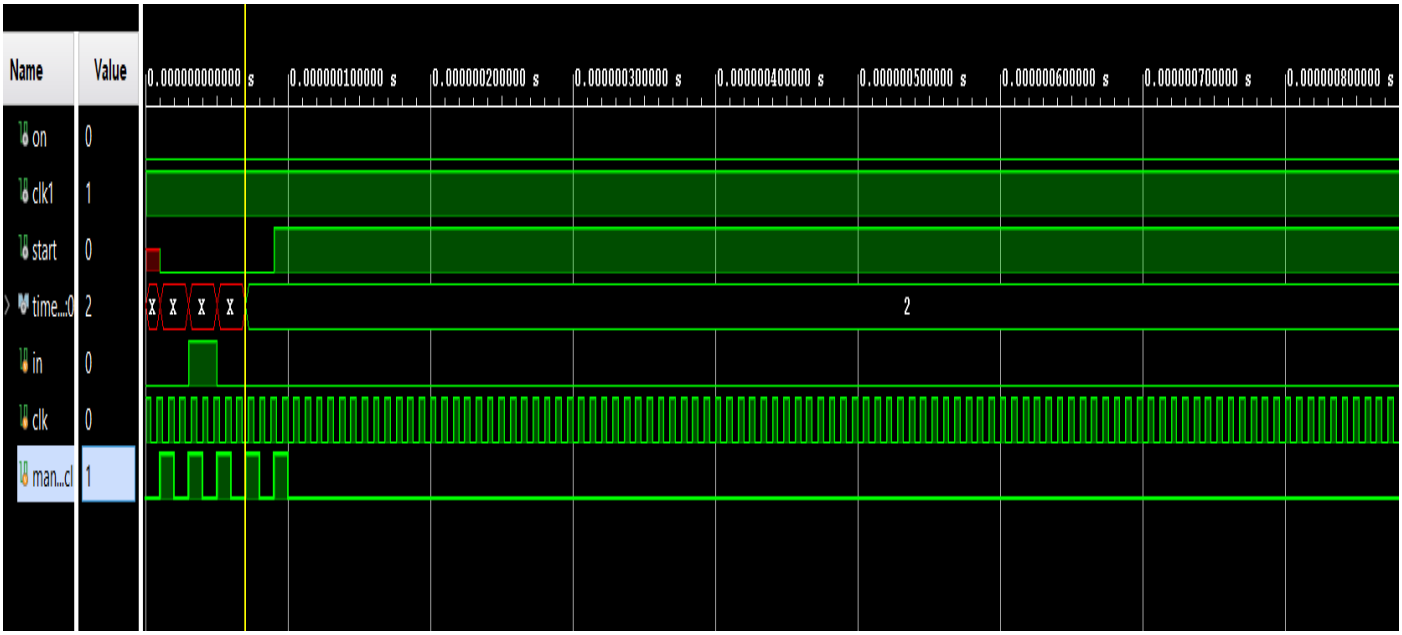
SYNTHESISED SCHEMATIC

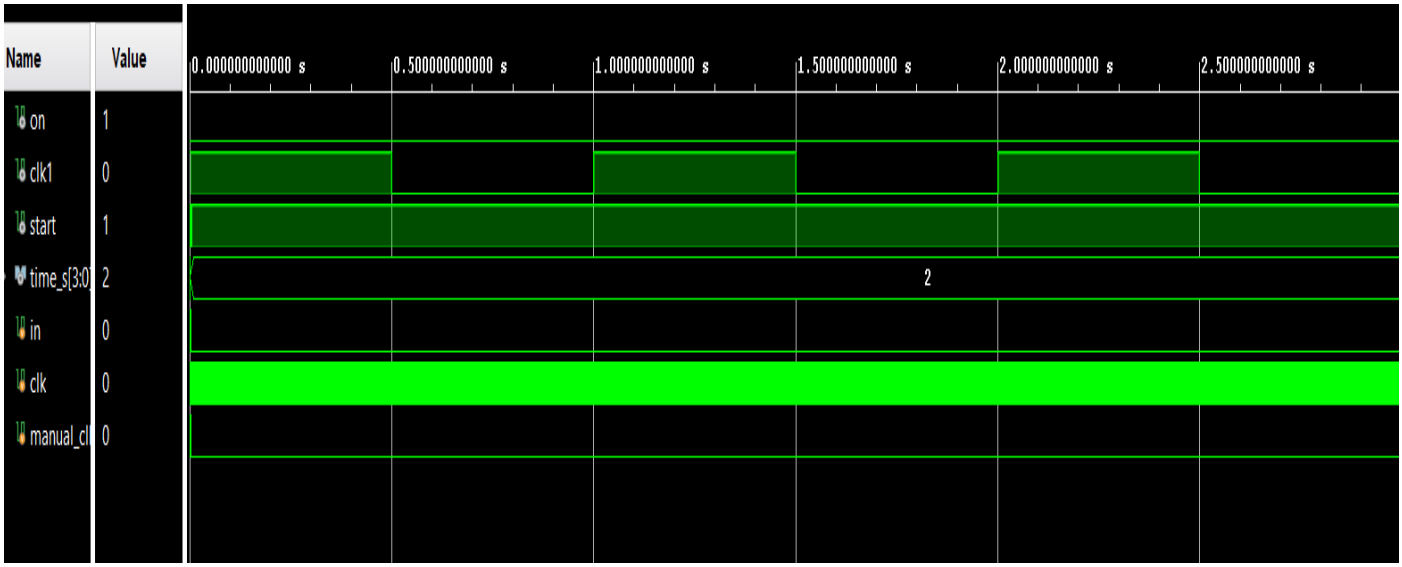


IMPLEMENTED SCHEMATIC

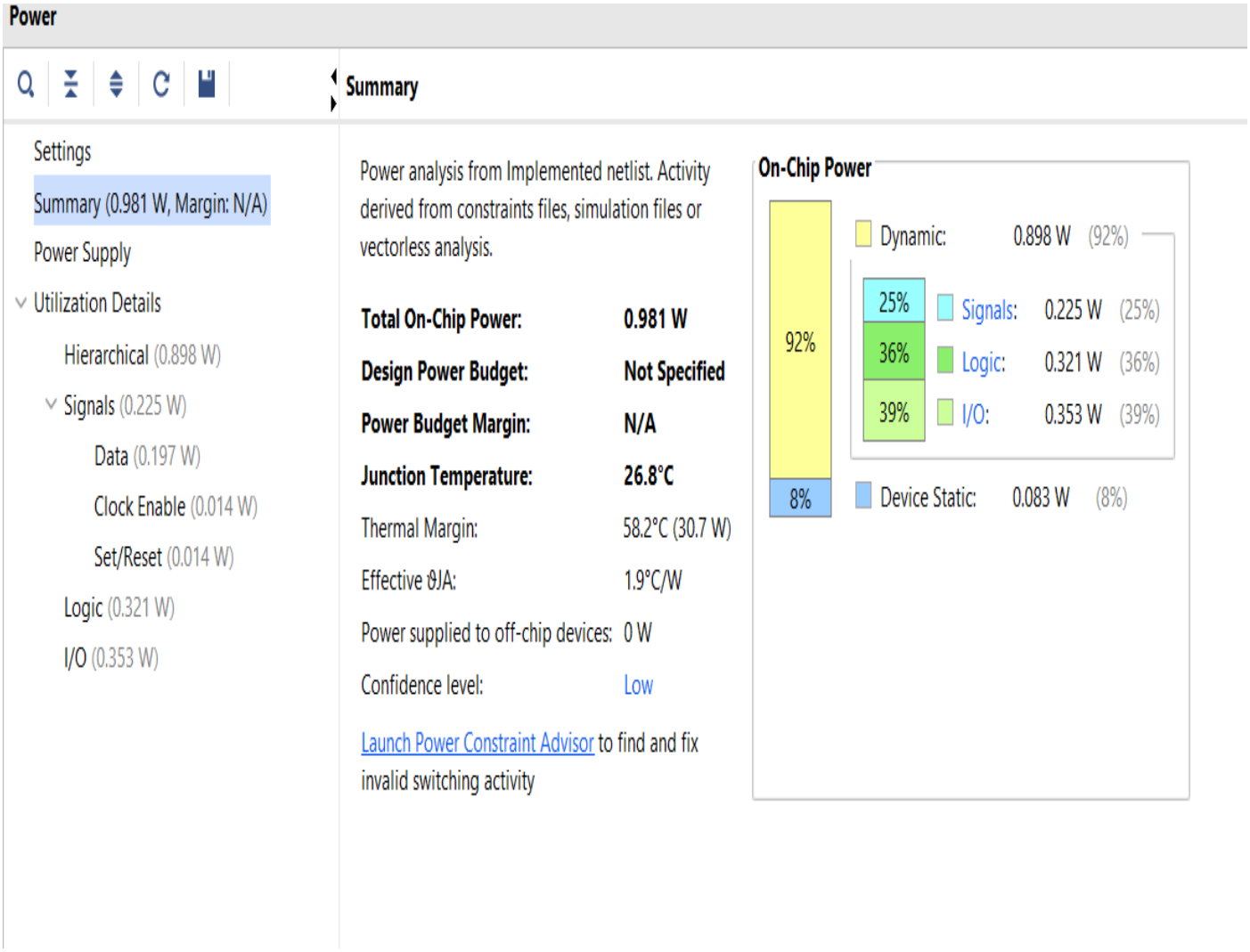


SIMULATION RESULTS

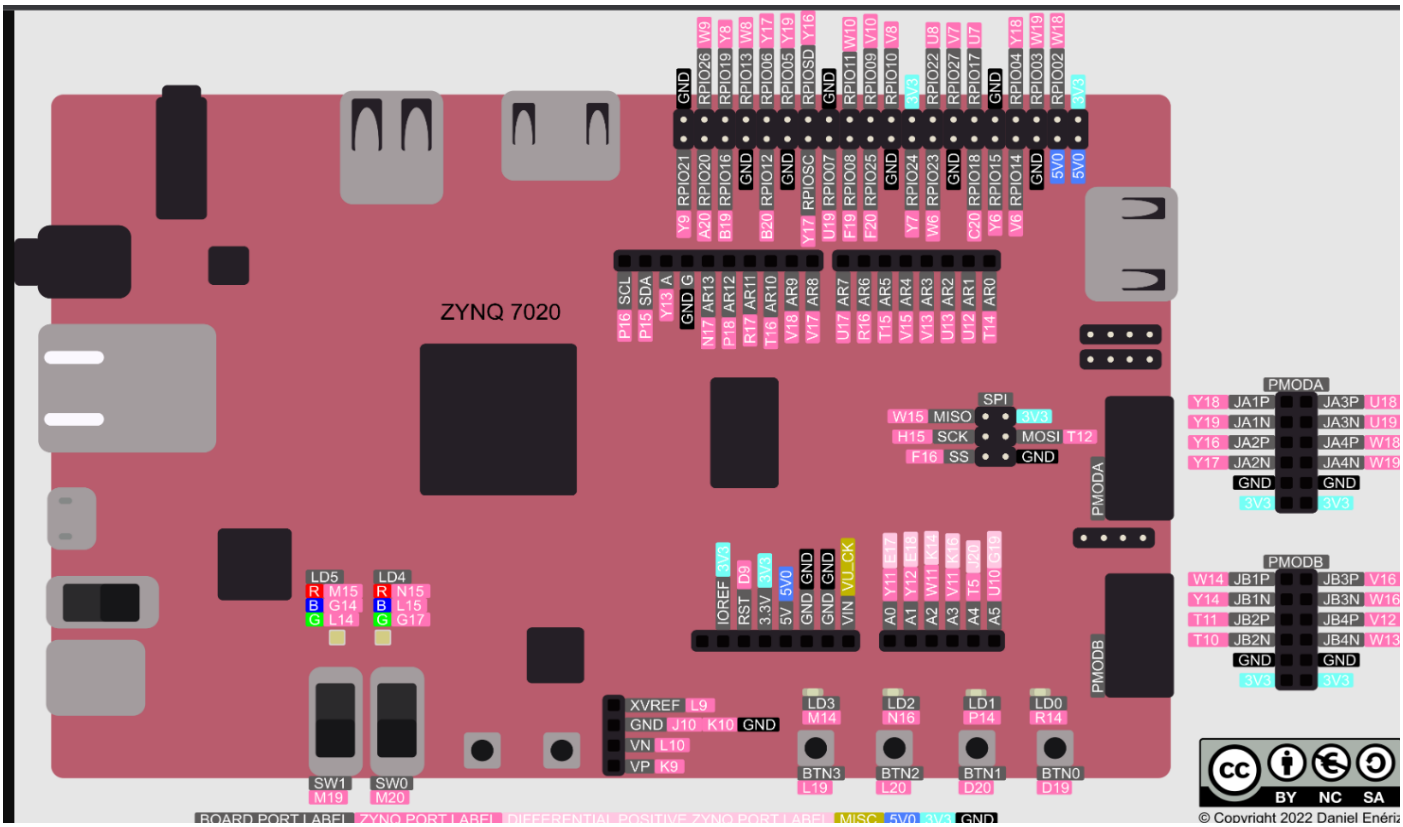




POWER ANALYSIS



PIN DIAGRAM



USER CONSTRAINT FILES

pins.xdc

C:/Users/DSP-LAB-PC7/Desktop/SC21B076/Ptoject_testing/Ptoject_testing.srcs/constrs_1/new/pins.xdc

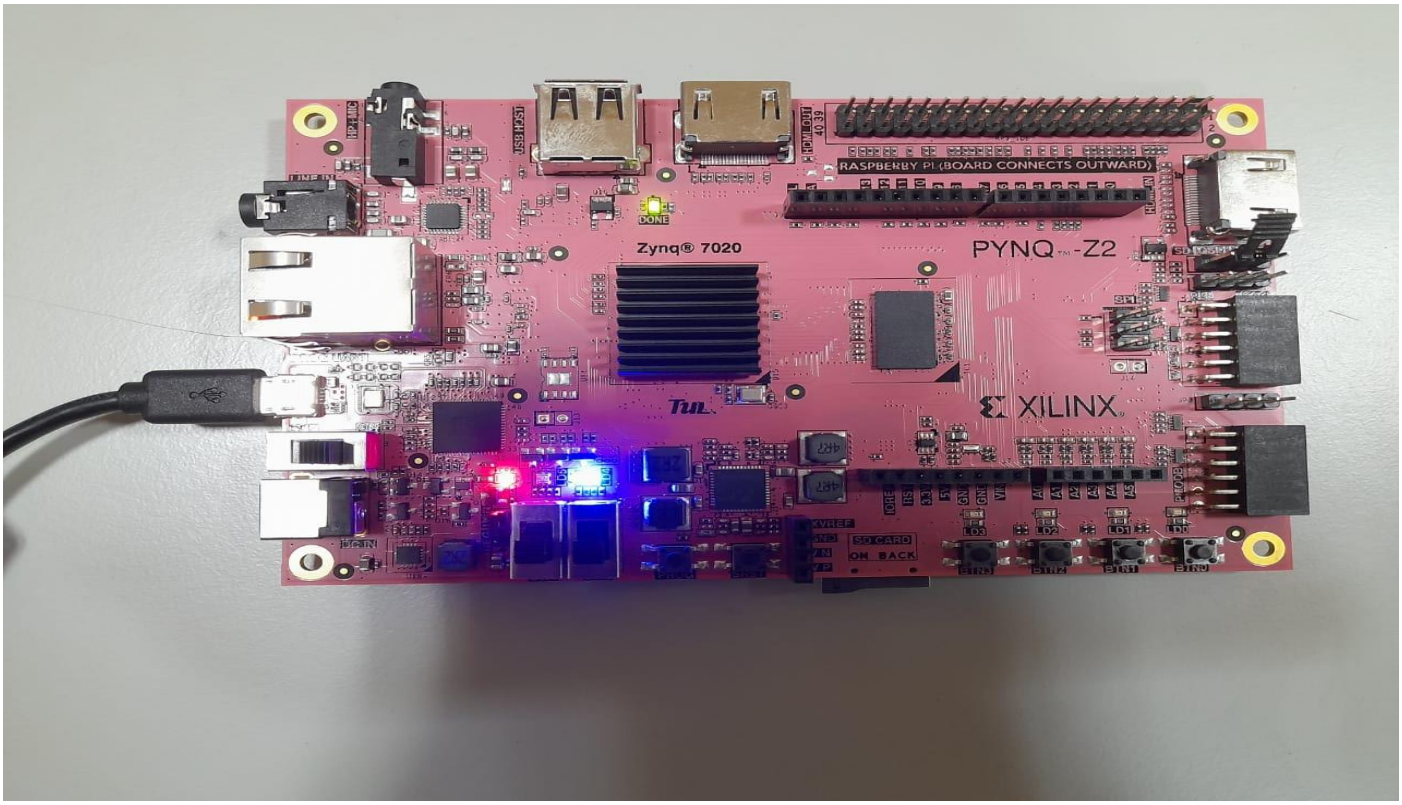
```
set_property IOSTANDARD LVCMOS33 [get_ports clk]
set_property IOSTANDARD LVCMOS33 [get_ports clk1]
set_property IOSTANDARD LVCMOS33 [get_ports in]
set_property IOSTANDARD LVCMOS33 [get_ports manual_clk]
set_property IOSTANDARD LVCMOS33 [get_ports on]
set_property IOSTANDARD LVCMOS33 [get_ports start]
set_property PACKAGE_PIN H16 [get_ports clk]
set_property PACKAGE_PIN L15 [get_ports clk1]
set_property PACKAGE_PIN M19 [get_ports in]
set_property PACKAGE_PIN M20 [get_ports manual_clk]
set_property PACKAGE_PIN L14 [get_ports start]
set_property PACKAGE_PIN M15 [get_ports on]
set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets manual_clk_IBUF]

set_property IOSTANDARD LVCMOS33 [get_ports {time_s[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {time_s[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {time_s[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {time_s[3]}]
set_property PACKAGE_PIN R14 [get_ports {time_s[0]}]
set_property PACKAGE_PIN P14 [get_ports {time_s[1]}]
set_property PACKAGE_PIN N16 [get_ports {time_s[2]}]
set_property PACKAGE_PIN M14 [get_ports {time_s[3]}]
```

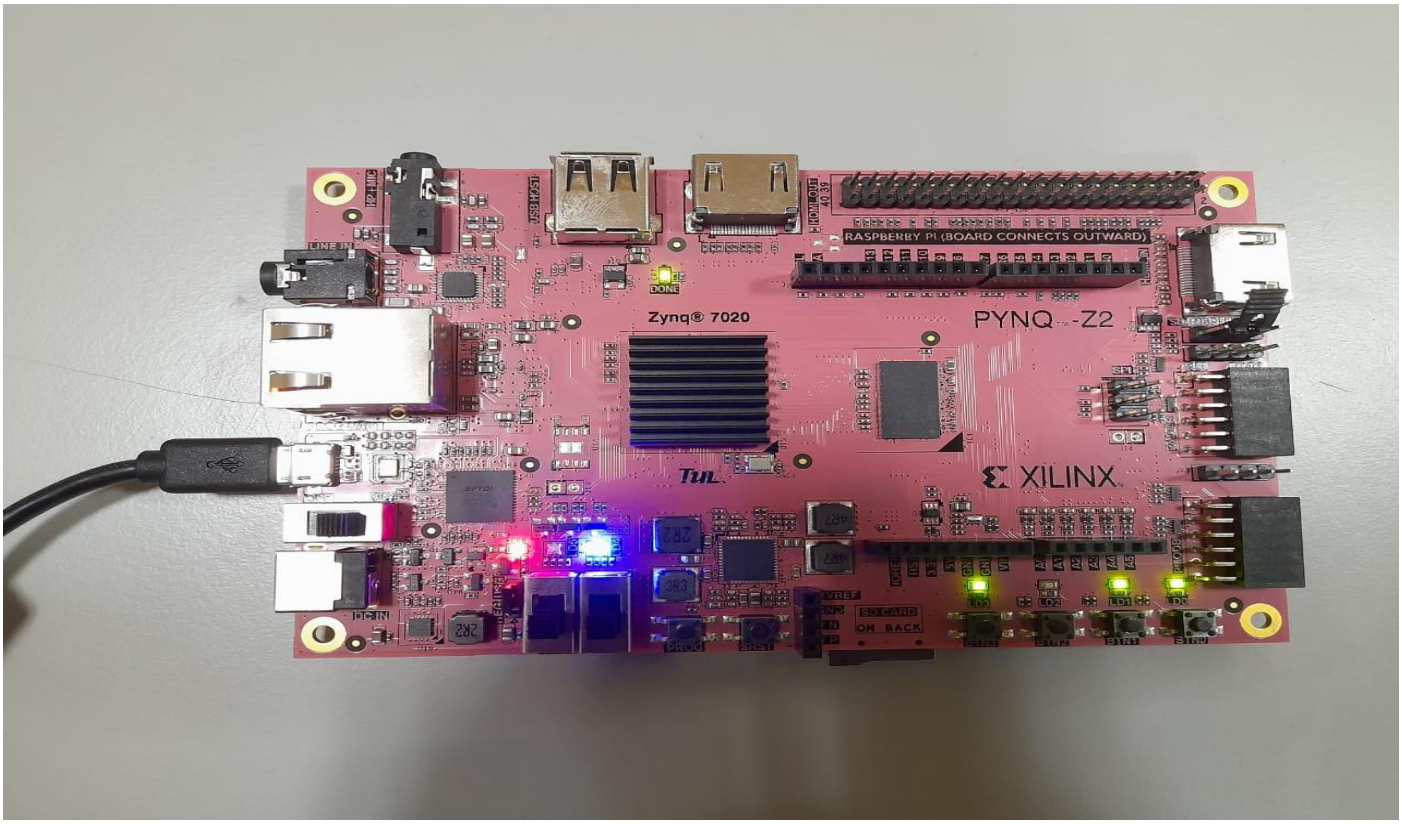
Find Results

Name	Direction	Board Part Pin	Board Part Interface	Interface	Neg Diff Pair	Package Pin	Fixed	Bank	I/O Std	Vcco	Vref	Drive Strength	Slew Type	Pull Type	Off-Chip Termination	IN_TERM	Partition Pin Location
clk	IN					H16	✓	35	LVCMOS33*	3.300				NONE	NONE		N/A
clk1	OUT					L15	✓	35	LVCMOS33*	3.300		12		NONE	FP_VTT_50		N/A
in	IN					M19	✓	35	LVCMOS33*	3.300				NONE	NONE		N/A
manual_clk	IN					M20	✓	35	LVCMOS33*	3.300				NONE	NONE		N/A
on	OUT					M15	✓	35	LVCMOS33*	3.300		12		NONE	FP_VTT_50		N/A
start	OUT					L14	✓	35	LVCMOS33*	3.300		12		NONE	FP_VTT_50		N/A
time_s[3]	OUT					M14	✓	35	LVCMOS33*	3.300		12		NONE	FP_VTT_50		N/A
time_s[2]	OUT					N16	✓	35	LVCMOS33*	3.300		12		NONE	FP_VTT_50		N/A
time_s[1]	OUT					P14	✓	34	LVCMOS33*	3.300		12		NONE	FP_VTT_50		N/A
time_s[0]	OUT					R14	✓	34	LVCMOS33*	3.300		12		NONE	FP_VTT_50		N/A

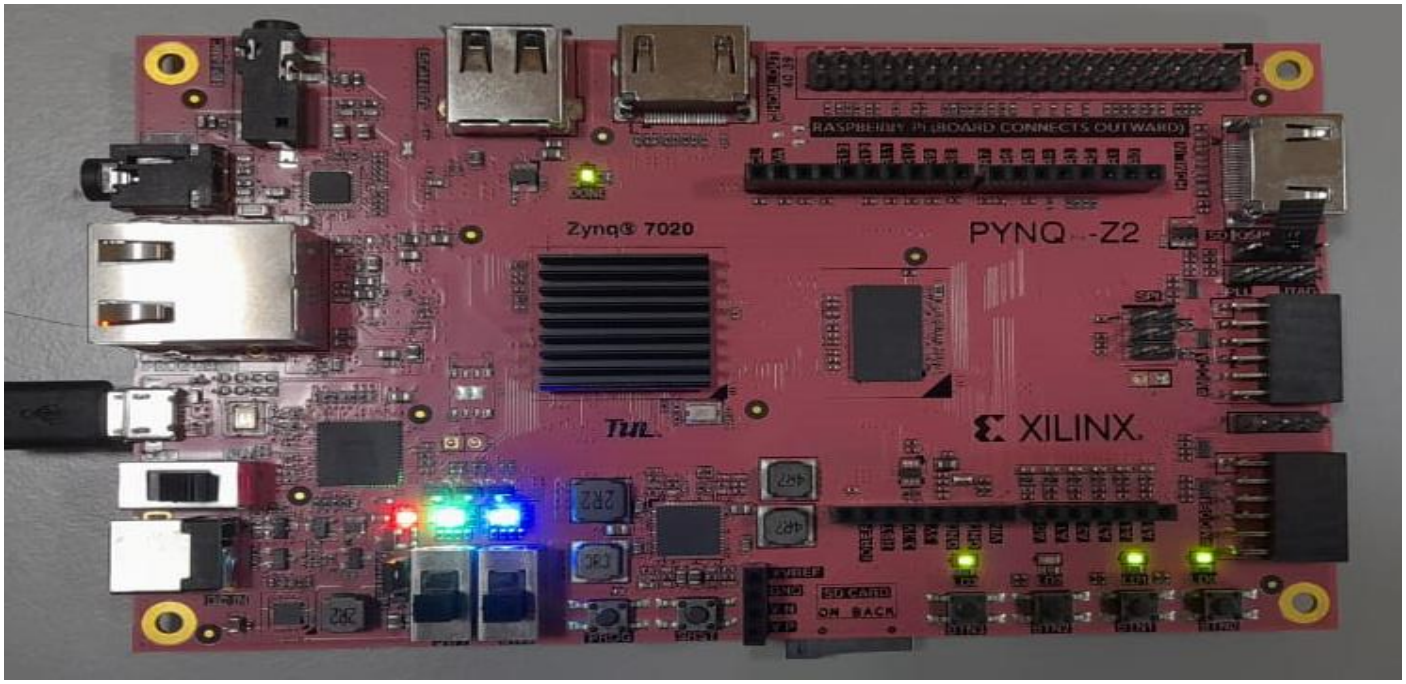
OUTPUT RESULT



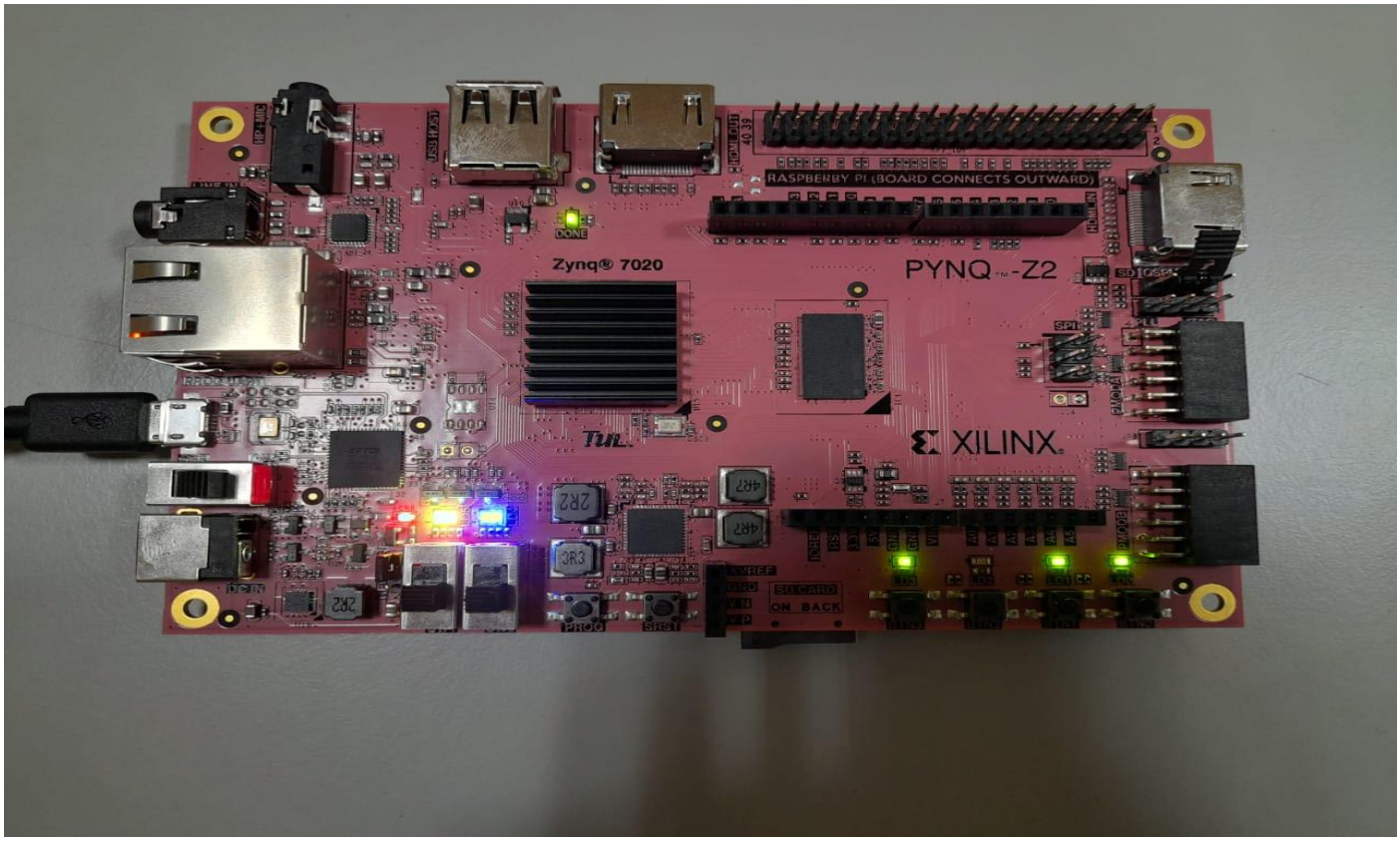
The inputs are given using the switches adjacent to the blue light.



The input given here is 1011. It sets the timer for 11 seconds.



The greenish line indicates that the clock has been set and counting has begun.



Once the counting is done by the clock mechanism, the greenish light turns into yellowish. The ideal color is meant to be different but due to interference between two light colors, the following output color is obtained, different but distinguishable.

INFERENCE

The design was successfully implemented on the PYNQ board using Vivado software and gave the desired output.