INDIAN INSTITUTE OF SPACE SCIENCE AND TECHNOLOGY

THIRUVANANTHAPURAM, KERALA

DIGITAL LAB PROJECT

DIGITAL CLOCK

BY:-

PAVITHRA ANIL(SC23B115)

ANJAN APARNA A B(SC23B087)

MAYANK PRATAP SINGH(SC23B110)

BTech ECE(Avionics)

2nd Year

Aim:
To make a digital clock that displays time in hours and minutes and has an option of setting time.

Approach: This is a 24 hour format digital clock that is counting seconds, minutes, hours, and displaying the current time on a 7 segment display. The clock utilizes a 125 MHz system clock signal from which a 1 Hz clock signal clk_1Hz is derived for time updates.

1. 1 Hz Clock Generation
A counter divides the 125 MHz system clock down to a 1 Hz signal (clk_1Hz). This signal is used to increment the time (seconds, minutes, and hours). The counter resets after every second, producing a 1 Hz pulse.

2. Time Setting and Reset:
The reset input initializes the clock to 00:00:00. The preset input, along with digit_set, allows users to set specific digits of the time:

2'b00: Set minutes and hours
2'b01: Clock minutes
2'b10: Clock hours
2'b11: Reset the clock
The logic makes the time increment correctly, roll over minutes at 59 and hours at 23.

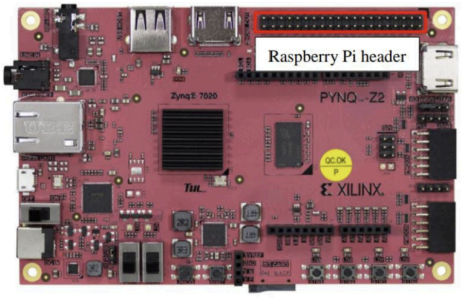
3. Counting Seconds, Minutes, and Hours:
The clock counts seconds every 1 Hz pulse. When seconds reach 59, minutes increment, and when minutes reach 59, hours increment. Hours roll over after 23.

4. Converting BCD to 7-Segment Display
Time values (hours, minutes, seconds) are converted to Binary Coded Decimal (BCD) format and then mapped to a 7-segment display. A BCD-to-7-segment module takes each 4-bit BCD digit and outputs the corresponding 7-bit signal to drive the display.

Key Features:
1 Hz Clock: Derived from a 125 MHz system clock.
Time Setting: User-controlled through digit_set and preset.
24-Hour Format: Automatic rollover for minutes and hours.
BCD to 7-Segment: Convert BCD digits to 7-segment display signals

Use of FPGA:

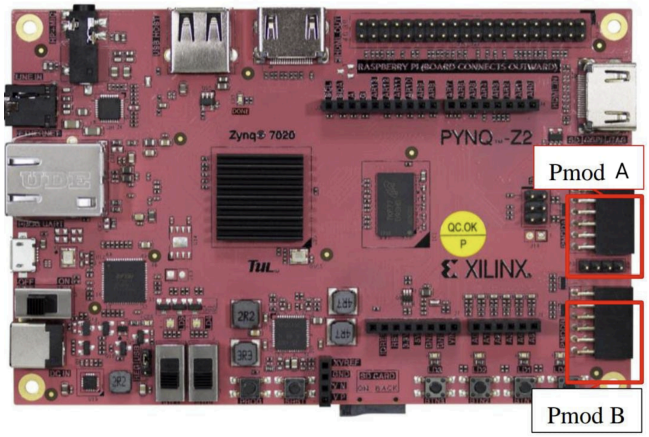
We used PMod A and Raspberry pi, LED



These pins can be used as GPIO, or to connect a standard Raspberry Pi peripheral.

G	W9	Y8	W8	Y7	Y6	Y16	G	W10	V10	V8	V	U8	V7	U7	G	V6	W19	W18	G
39	37	35	33	31	29	27	25	23	21	19	17	15	13	11	9	7	5	3	1
40	38	36	34	32	30	28	26	24	22	20	18	16	14	12	10	8	6	4	2
Y9	A20	B19	G	B20	G	Y17	F20	F19	U19	G	U18	W6	G	C20	Y19	Y18	G	V	V

G	Ground
V	3.3V
V	5V
...	Raspberry Pi header pin number
...	Zynq Pin



Verilog code

Main module

```
`timescale 1ns /1ps
//////////////////////////////////////////////////////////////////////////////////

module digital_clock (
    input clk, // Clock input
    input reset, // Reset input
    input [1:0] digit_set,
    input preset,
    output [6:0] HT,
    output [6:0] HU,
    output [6:0] MT,
    output [6:0] MU
);
    reg [3:0] hours_tens,hours_units,minutes_tens,minutes_units;
         reg clk_1Hz ;
```

```
     reg [3:0] Hours_tens, Hours_units,  Minutes_tens,
Minutes_units, seconds_tens, seconds_units;
    // Generate 1 Hz clock
    reg [26:0] counter_1s = 0;
    parameter integer N = 125000000;  // Assuming f_clk = 125 MHz

    always @(posedge clk or posedge reset) begin
        if (reset) begin
            counter_1s <= 0;
            clk_1Hz <= 0;
        end else if (counter_1s == N - 1) begin
            counter_1s <= 0;
            clk_1Hz <= ~clk_1Hz;
        end else begin
            counter_1s <= counter_1s + 1;
        end
    end

//Code to set time
always @(reset,preset) begin

    if (reset) begin
        minutes_units<=0;
        minutes_tens<=0;
        hours_units<=0;
        hours_tens<=0; end
    else
    if (preset) begin
        case(digit_set)
        2'b00: begin
                if(minutes_units==4'd9) begin
                    if(minutes_tens==4'd5) begin

if(hours_tens==4'd2&&hours_units==4'd3)begin
                            minutes_units<=0;
                            minutes_tens<=0;
                            hours_units<=0;
                            hours_tens<=0; end
```

```
                        else if(hours_units==4'd9) begin
                           hours_tens<=hours_tens+1;
                           hours_units<=0;
                           minutes_tens=0;
                           end
                       else begin
                       minutes_units=0;
                       minutes_tens= 0;
                       hours_units=hours_units+1; end
                   end
                 else begin
                       minutes_units=0;
                       minutes_tens=minutes_tens+1; end
               end
               else minutes_units=minutes_units+1; end
         2'b01:
                   if(minutes_tens==4'd5) begin

if(hours_tens==4'd2&&hours_units==4'd3)begin
                           minutes_units<=0;
                           minutes_tens<=0;
                           hours_units<=0;
                           hours_tens<=0; end
                        else if(hours_units==4'd9) begin
                           hours_tens<=hours_tens+1;
                           hours_units<=0;
                           minutes_tens=0;
                           end
                       else begin
                       hours_units<=hours_units+1;
                       minutes_tens<=0; end
                       end
                   else
                       minutes_tens=minutes_tens+1;

         2'b10:
               if(hours_tens==4'd2&&hours_units==4'd3)begin
                       minutes_units<=0;
```

```
                minutes_tens<=0;
                hours_units<=0;
                hours_tens<=0; end
        else if(hours_units==4'd9) begin
                hours_tens<=hours_tens+1;
                hours_units<=0;
             end
        else
              hours_units<=hours_units+1;

 2'b11:
        if(hours_tens>=4'd2) begin
                minutes_units<=0;
                minutes_tens<=0;
                hours_units<=0;
                hours_tens<=0; end
            else
                hours_tens=hours_tens+1;
 default:  begin minutes_units<=minutes_units;
           minutes_tens<=minutes_tens;
           hours_units<=hours_units;
           hours_tens<=hours_tens;
           end

 endcase
end
 /*Hours_tens<=hours_tens;
 Hours_units<= hours_units;
 Minutes_tens<=minutes_tens;
 Minutes_units<=minutes_units;*/
end

reg [5:0] seconds_count;
reg [5:0] minutes_count;
reg [4:0] hours_count;
```

```
always @(posedge clk_1Hz or posedge reset or posedge preset )
begin
    if (preset)begin
    if (reset) begin
        // Reset all counts to zero
        seconds_count <= 0;
        minutes_count <= 0;
        hours_count <= 0;
    end
    else begin
    seconds_count <= 0;
    minutes_count <= ((minutes_tens*4'd10)+(minutes_units));
    hours_count <= (hours_tens*4'd10)+(hours_units);
    end
    end
    else begin
    if (reset) begin
        // Reset all counts to zero
        seconds_count <= 0;
        minutes_count <= 0;
        hours_count <= 0;
    end

    /*else if(set)begin
    seconds_count <= 0;
    minutes_count <= ((minutes_tens*4'd10)+(minutes_units));
    hours_count <= (hours_tens*4'd10)+(hours_units);
    end*/

    else begin
        // Increment the seconds count
        if (seconds_count < 6'd59) begin
            seconds_count <= seconds_count + 1;
        end else begin
            seconds_count <= 0;
            // Increment the minutes count
            if (minutes_count < 6'd59) begin
                minutes_count <= minutes_count + 1;
```

```
            end else begin
                minutes_count <= 0;
                // Increment the hours count
                if (hours_count < 5'd23) begin
                    hours_count <= hours_count + 1;
                end else begin
                    hours_count <= 0;
                end
            end
        end
    end end
end

// Update the BCD outputs
always @((clk |reset) & ~preset) begin
    seconds_units = seconds_count % 4'd10;
    seconds_tens = seconds_count / 4'd10;
    Minutes_units = minutes_count % 4'd10;
    Minutes_tens = minutes_count / 4'd10;
    Hours_units = hours_count % 4'd10;
    Hours_tens = hours_count / 4'd10;
end


BCDtoSevenSegmentDisplay M1(Minutes_units,MU);
BCDtoSevenSegmentDisplay M2(Minutes_tens,MT);
BCDtoSevenSegmentDisplay H1(Hours_units,HU);
BCDtoSevenSegmentDisplay H2(Hours_tens,HT);


endmodule
```

<u>BCD to Seven Segment Display</u>

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////
/////////////////
// Company:
// Engineer:Anjan Aparna A B
```

```
//
// Create Date: 24.11.2024 11:33:18
// Design Name:
// Module Name: BCDtoSevenSegmentDisplay
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////
////////////////


module BCDtoSevenSegmentDisplay(
    input [3:0] Digit,
    output reg [6:0] O
    );
    always@(*) begin
    case(Digit)
    4'd0:O=7'b1111110;
    4'd1:O=7'b0110000;
    4'd2:O=7'b1101101;
    4'd3:O=7'b1111001;
    4'd4:O=7'b0110011;
    4'd5:O=7'b1011011;
    4'd6:O=7'b1011111;
    4'd7:O=7'b1110000;
    4'd8:O=7'b1111111;
    4'd9:O=7'b1111011;
    4'd10,4'd11,4'd12,4'd13,4'd14,4'd15:O=7'b1111110;
    default:O=7'b1111110;
    endcase
```

```
    end
endmodule
```

Testbench Code

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 23.11.2024 12:01:38
// Design Name:
// Module Name: dg_clk_tb
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module dg_clk_tb();
// Inputs
reg clk;
reg reset;
reg preset;
reg [1:0] digit_set;

// Outputs
/*wire clk_1Hz ;
wire [3:0] Hours_tens;
```

```
wire [3:0] Hours_units;
wire [3:0] Minutes_tens;
wire [3:0] Minutes_units;
wire [3:0] seconds_tens;
wire [3:0] seconds_units;*/
   wire [6:0] HT;
    wire [6:0] HU;
   wire [6:0] MT;
   wire [6:0] MU;
// Instantiate the Unit Under Test (UUT)
digital_clock uut (
    .clk(clk),
    .reset(reset),
    .preset(preset),
    .digit_set(digit_set),
     .HT(HT),
    .HU(HU),
     .MT(MT),
     .MU(MU)
);

initial begin
    // Initialize Inputs
    clk = 0;
    reset = 1;
    preset=0;
    digit_set=2'b00;
    // Wait 100 ns for global reset to finish
    #100
    preset=0;
    #10
    reset = 0;
    preset=1;
    digit_set=2'b00;
    #10
    preset=0;
    #10
    preset=1;
```

```
digit_set=2'b00;
#10
preset=0;
#10
preset=1;
digit_set=2'b00;
#10
preset=0;
#10
preset=1;
digit_set=2'b00;
#10
preset=0;
#10
preset=1;
digit_set=2'b00;
#10
preset=0;
#10
preset=1;
digit_set=2'b00;
#10
preset=0;
#10
preset=1;
digit_set=2'b00;
#10
preset=0;
#10
preset=1;
digit_set=2'b01;
#10
preset=0;
```

```
#10
preset=1;
digit_set=2'b01;
#10
preset=0;
#10
preset=1;
digit_set=2'b01;
#10
preset=0;
#10
preset=1;
digit_set=2'b01;
#10
preset=0;
#10
preset=1;
digit_set=2'b01;
#10
preset=0;
#10
preset=1;
digit_set=2'b10;
#10
preset=0;
#10
preset=1;
digit_set=2'b10;
#10
preset=0;
#10
preset=1;
digit_set=2'b10;
#10
preset=0;
#10
preset=1;
digit_set=2'b11;
```

```
    #10
    preset=0;
    #10
    preset=1;
    digit_set=2'b11;
    #10
    preset=0;
    #10

    preset=0;

    // Wait for some time to observe the outputs
    #10000000; // You can adjust this delay to simulate a longer
time period

    // Reset again
   // reset = 1;

    reset = 0;

    // Wait for some more time
    #1000000; // Adjust this delay as needed

    // Finish simulation
    $finish;
end

always begin
    #1 clk = ~clk; // Generate a clock with a period of 2 units
end


endmodule
```

Constraint file

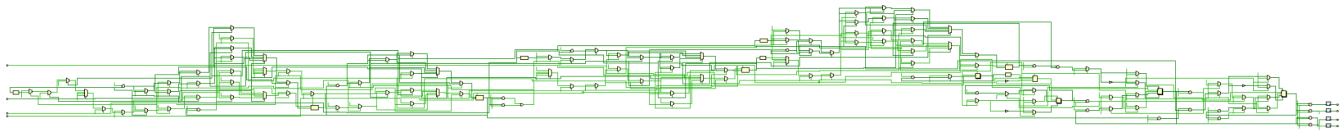
```
set_property -dict { PACKAGE_PIN H16   IOSTANDARD LVCMOS33 }
[get_ports {clk}]; #IO_L13P_T2_MRCC_35 Sch=sysclk
```

```
set_property -dict { PACKAGE_PIN M20   IOSTANDARD LVCMOS33 }
[get_ports { digit_set[0]}]; #IO_L7N_T1_AD2N_35 Sch=sw[0]
set_property -dict { PACKAGE_PIN M19   IOSTANDARD LVCMOS33 }
[get_ports { digit_set[1]  }]; #IO_L7P_T1_AD2P_35 Sch=sw[1]

set_property -dict { PACKAGE_PIN D19   IOSTANDARD LVCMOS33 }
[get_ports { preset }]; #IO_L4P_T0_35 Sch=btn[0]
set_property -dict { PACKAGE_PIN D20   IOSTANDARD LVCMOS33 }
[get_ports { reset}]; #IO_L4N_T0_35 Sch=btn[1]

set_property -dict { PACKAGE_PIN R14   IOSTANDARD LVCMOS33 }
[get_ports { MU[0] }]; #IO_L6N_T0_VREF_34 Sch=led[0]
set_property -dict { PACKAGE_PIN P14   IOSTANDARD LVCMOS33 }
[get_ports { MU[1] }]; #IO_L6P_T0_34 Sch=led[1]
set_property -dict { PACKAGE_PIN N16   IOSTANDARD LVCMOS33 }
[get_ports { MU[2] }]; #IO_L21N_T3_DQS_AD14N_35 Sch=led[2]
set_property -dict { PACKAGE_PIN M14   IOSTANDARD LVCMOS33 }
[get_ports { MU[3] }]; #IO_L23P_T3_35 Sch=led[3]
set_property -dict { PACKAGE_PIN U18   IOSTANDARD LVCMOS33 }
[get_ports { MU[4] }]; #IO_L12P_T1_MRCC_34 Sch=ja_p[3]
set_property -dict { PACKAGE_PIN U19   IOSTANDARD LVCMOS33 }
[get_ports { MU[5] }]; #IO_L12N_T1_MRCC_34 Sch=ja_n[3]
set_property -dict { PACKAGE_PIN W18   IOSTANDARD LVCMOS33 }
[get_ports { MU[6] }]; #IO_L22P_T3_34 Sch=ja_p[4]


#set_property -dict { PACKAGE_PIN W18   IOSTANDARD LVCMOS33 }
[get_ports { MU[0] }]; #IO_L22P_T3_34 Sch=rpio_02_r
#set_property -dict { PACKAGE_PIN W19   IOSTANDARD LVCMOS33 }
[get_ports { MU[1] }]; #IO_L22N_T3_34 Sch=rpio_03_r
set_property -dict { PACKAGE_PIN T14   IOSTANDARD LVCMOS33 }
[get_ports { HU[2] }]; #IO_L5P_T0_34 Sch=ar[0]
#set_property -dict { PACKAGE_PIN Y18   IOSTANDARD LVCMOS33 }
[get_ports { MU[3] }]; #IO_L17P_T2_34 Sch=rpio_04_r
#set_property -dict { PACKAGE_PIN Y19   IOSTANDARD LVCMOS33 }
[get_ports { MU[4] }]; #IO_L17N_T2_34 Sch=rpio_05_r
```

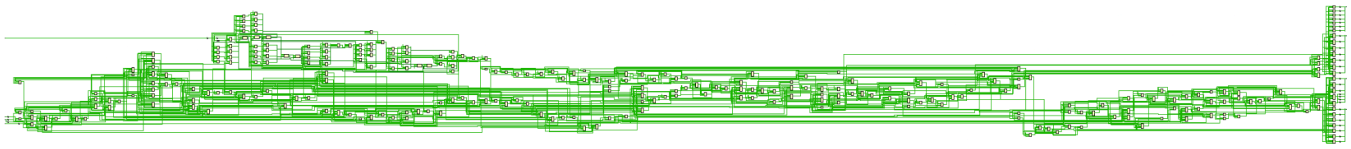
```
#set_property -dict { PACKAGE_PIN U18   IOSTANDARD LVCMOS33 }
[get_ports { MU[5] }]; #IO_L22P_T3_13 Sch=rpio_06_r
#set_property -dict { PACKAGE_PIN U19   IOSTANDARD LVCMOS33 }
[get_ports {MU[6] }]; #IO_L12P_T1_MRCC_34 Sch=rpio_07_r
set_property -dict { PACKAGE_PIN F19   IOSTANDARD LVCMOS33 }
[get_ports { MT[0] }]; #IO_L12N_T1_MRCC_34 Sch=rpio_08_r
set_property -dict { PACKAGE_PIN V10   IOSTANDARD LVCMOS33 }
[get_ports { MT[1] }]; #IO_L21N_T3_DQS_13 Sch=rpio_09_r
set_property -dict { PACKAGE_PIN V8    IOSTANDARD LVCMOS33 }
[get_ports { MT[2] }]; #IO_L15P_T2_DQS_13 Sch=rpio_10_r
set_property -dict { PACKAGE_PIN W10   IOSTANDARD LVCMOS33 }
[get_ports { MT[3] }]; #IO_L16P_T2_13 Sch=rpio_11_r
set_property -dict { PACKAGE_PIN B20   IOSTANDARD LVCMOS33 }
[get_ports { MT[4] }]; #IO_L1N_T0_AD0N_35 Sch=rpio_12_r
set_property -dict { PACKAGE_PIN W8    IOSTANDARD LVCMOS33 }
[get_ports { MT[5] }]; #IO_L15N_T2_DQS_13 Sch=rpio_13_r
set_property -dict { PACKAGE_PIN V6    IOSTANDARD LVCMOS33 }
[get_ports { MT[6] }]; #IO_L22P_T3_13 Sch=rpio_14_r
set_property -dict { PACKAGE_PIN Y6    IOSTANDARD LVCMOS33 }
[get_ports { HU[0] }]; #IO_L13N_T2_MRCC_13 Sch=rpio_15_r
set_property -dict { PACKAGE_PIN B19   IOSTANDARD LVCMOS33 }
[get_ports { HU[1] }]; #IO_L2P_T0_AD8P_35 Sch=rpio_16_r
#set_property -dict { PACKAGE_PIN U7    IOSTANDARD LVCMOS33 }
[get_ports { MU[2] }]; #IO_L11P_T1_SRCC_13 Sch=rpio_17_r
set_property -dict { PACKAGE_PIN C20   IOSTANDARD LVCMOS33 }
[get_ports { HU[3] }]; #IO_L1P_T0_AD0P_35 Sch=rpio_18_r
set_property -dict { PACKAGE_PIN Y8    IOSTANDARD LVCMOS33 }
[get_ports { HU[4] }]; #IO_L14N_T2_SRCC_13 Sch=rpio_19_r
set_property -dict { PACKAGE_PIN A20   IOSTANDARD LVCMOS33 }
[get_ports { HU[5] }]; #IO_L2N_T0_AD8N_35 Sch=rpio_20_r
set_property -dict { PACKAGE_PIN Y9    IOSTANDARD LVCMOS33 }
[get_ports { HU[6] }]; #IO_L14P_T2_SRCC_13 Sch=rpio_21_r
set_property -dict { PACKAGE_PIN U8    IOSTANDARD LVCMOS33 }
[get_ports { HT[0] }]; #IO_L17N_T2_13 Sch=rpio_22_r
set_property -dict { PACKAGE_PIN W6    IOSTANDARD LVCMOS33 }
[get_ports { HT[1] }]; #IO_IO_L22N_T3_13 Sch=rpio_23_r
set_property -dict { PACKAGE_PIN Y7    IOSTANDARD LVCMOS33 }
[get_ports { HT[2] }]; #IO_L13P_T2_MRCC_13 Sch=rpio_24_r
```

```
set_property -dict { PACKAGE_PIN F20   IOSTANDARD LVCMOS33 }
[get_ports { HT[3] }]; #IO_L15N_T2_DQS_AD12N_35 Sch=rpio_25_r
set_property -dict { PACKAGE_PIN W9    IOSTANDARD LVCMOS33 }
[get_ports { HT[4] }]; #IO_L16N_T2_13 Sch=rpio_26_r
set_property -dict { PACKAGE_PIN W19   IOSTANDARD LVCMOS33 }
[get_ports { HT[5] }]; #IO_L22N_T3_34 Sch=ja_n[4]
set_property -dict { PACKAGE_PIN W14   IOSTANDARD LVCMOS33 }
[get_ports { HT[6] }]; #IO_L8P_T1_34 Sch=jb_p[1]
```

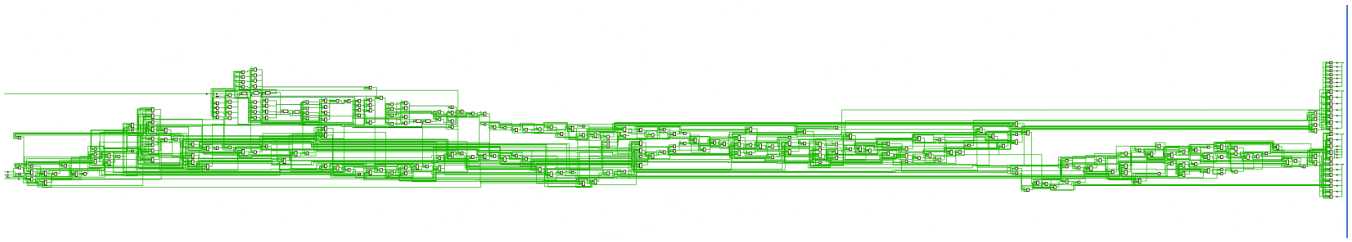
Schematic



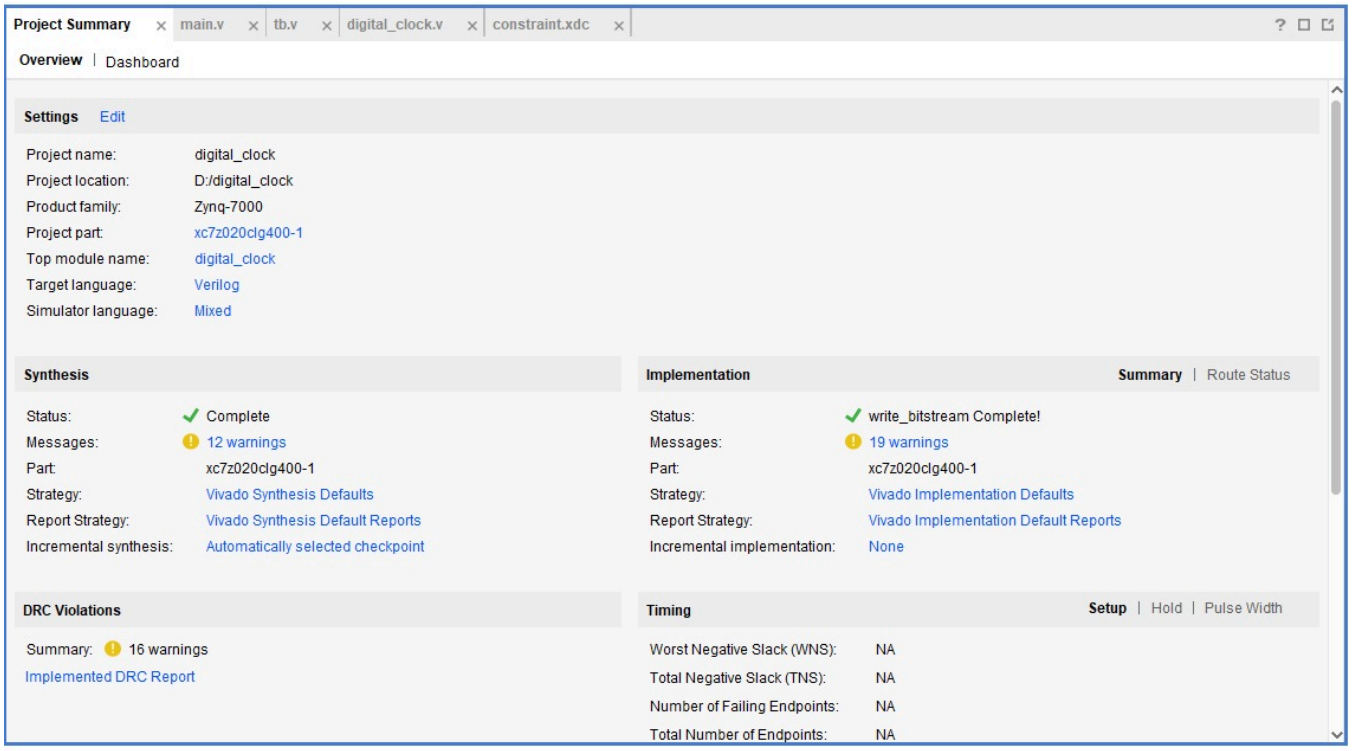
Synthesised Schematic

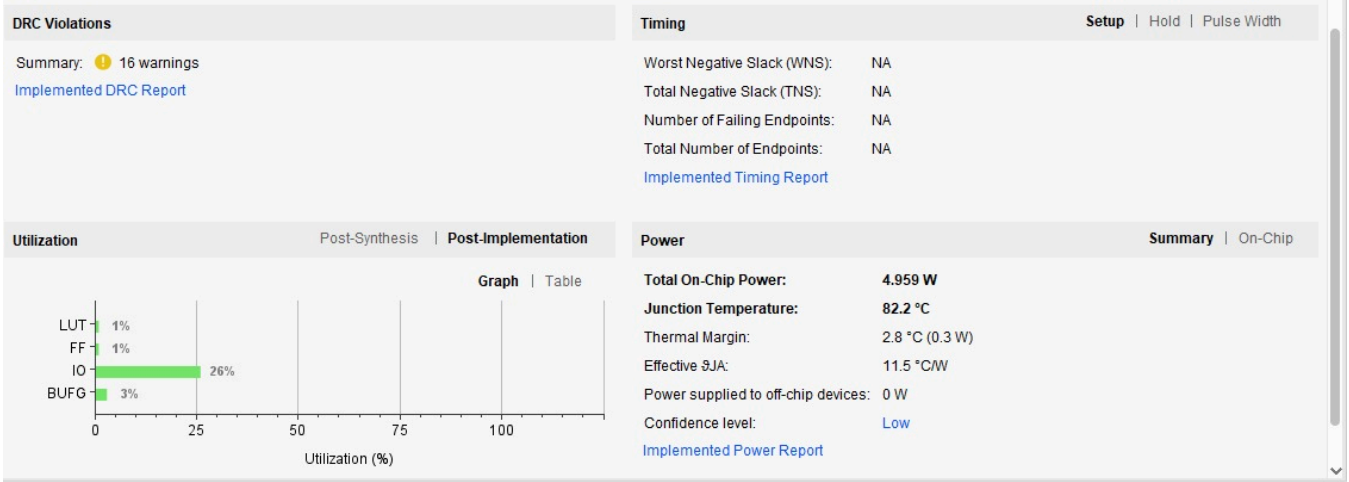


Implemented Schematic



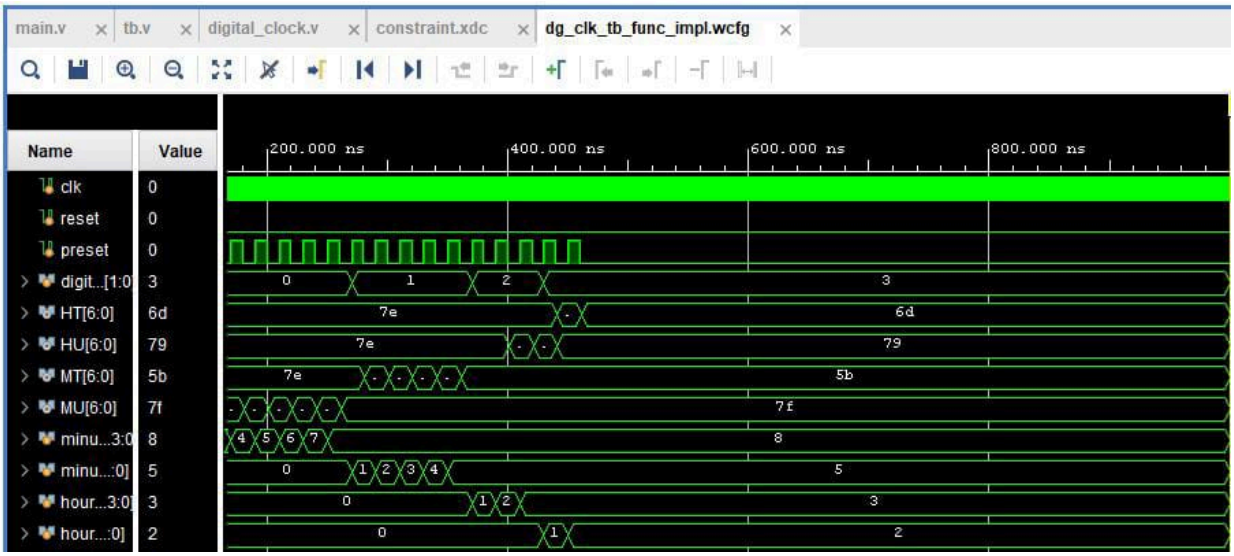
Power Analysis



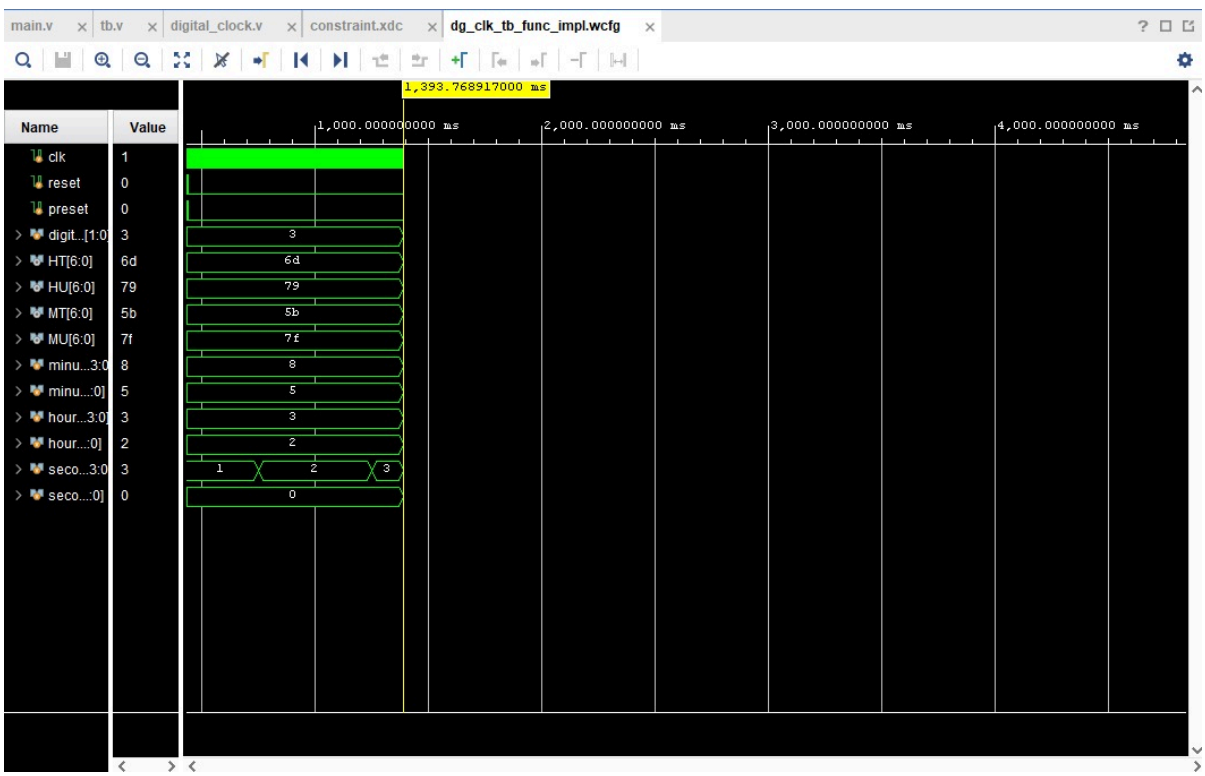


Simulation

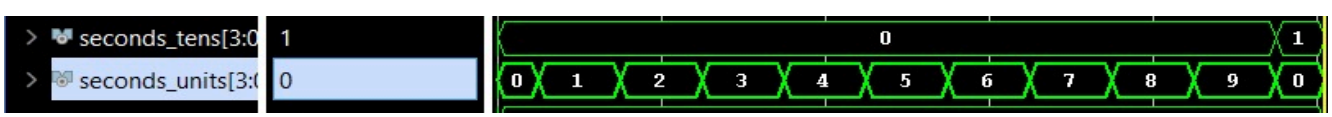
Behavioural Simulation



Behavioral simulation of the clock showing the seven segment outputs with time being set as 23:58

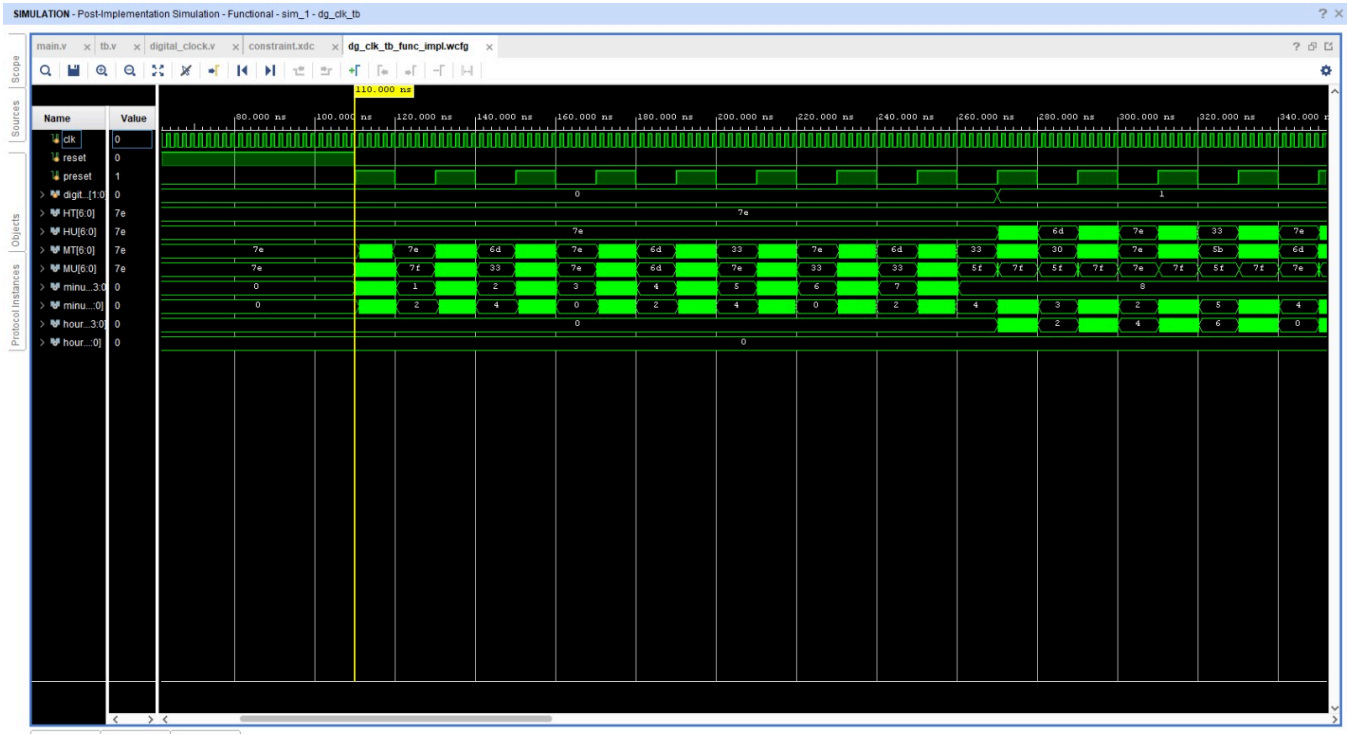


Behavioral simulation : time -23:58:03

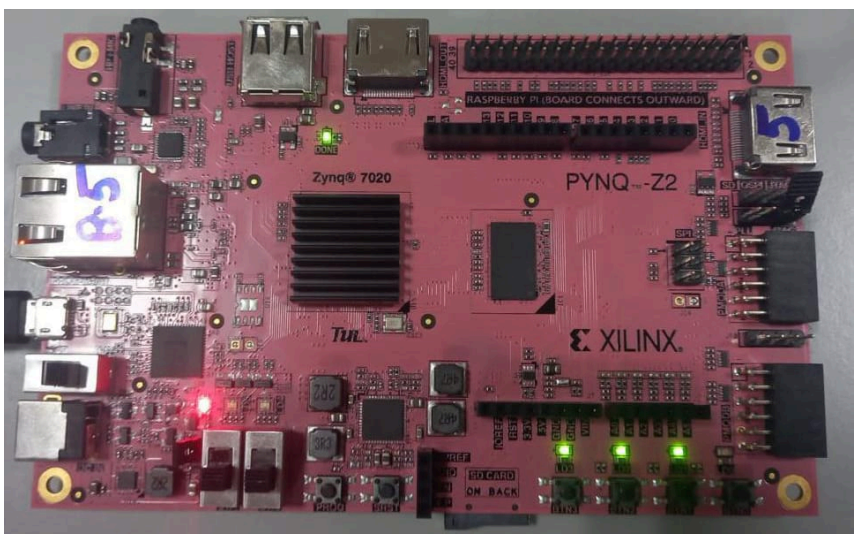


Behavioral simulation showing 10s

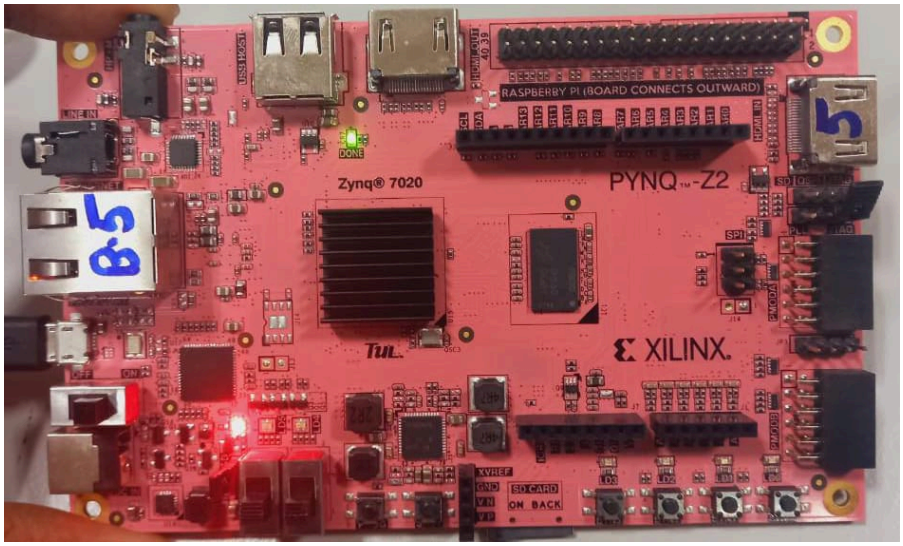
Post Implementation Behavioural Simulation



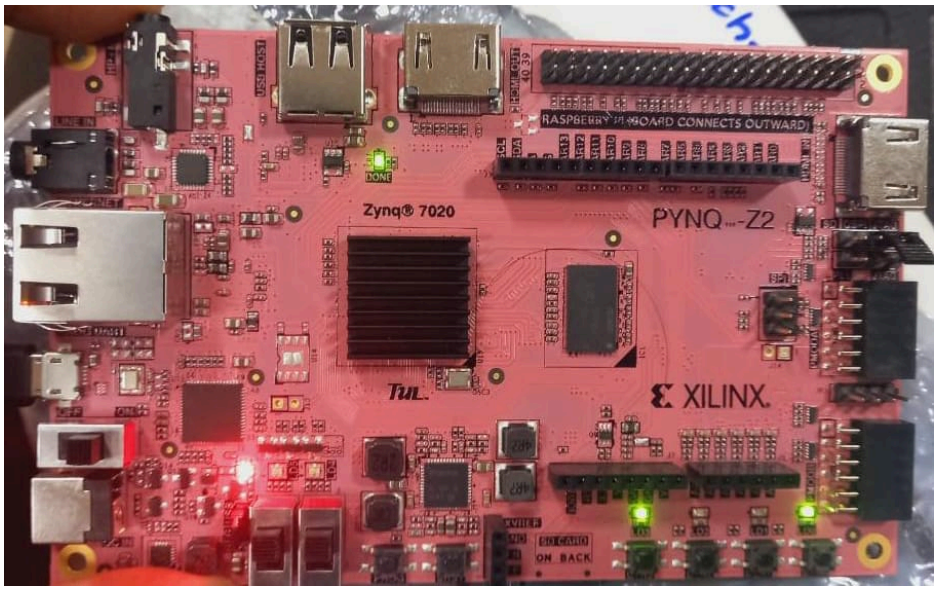
Board output pic



1110 -> 0 minutes



0000->1 minute



1001 ->2 minutes