CS-431 – Digital System Design (DSD)

Complex Engineering Activity (CEP)

Topic: Temperature Sensor using Artix-7 FGPA

Group Member Names along with Seat Numbers and Practical Group:

Aleena Yameen – CS-20047 – G3

Safia Faiz – CS-20058 – G2

Syed Muhammad Mubashir Rizvi – CS-20071 – G2

Sunyah Faisal – CS-20082 – G3

Submitted To:

Ms. Syeda Ramish Fatima

DEPARTMENT OF COMPUTER & INFORMATION SYSTEMS ENGINEERING
BACHELORS IN COMPUTER SYSTEMS ENGINEERING
Course Code: CS-431
Course Title: Digital System Design
Complex Engineering Problem
BE Batch 2019, Fall Semester 2022
CEA Grading Rubric
TERM PROJECT

CRITERIA AND SCALES			
Criterion 1: Does the FPGA application meet the desired specifications and produce the desired outputs? [CPA 1, CPA 2]			
1	2	3	4
The application does not meet the desired specifications and is producing incorrect outputs.	The application partially meets the desired specifications and is producing incorrect or partially correct outputs.	The application meets the desired specifications but is producing incorrect or partially correct outputs.	The application meets all the desired specifications and is producing correct outputs.
Criterion 2: What is the student's level of confidence with handling of equipment and tools? [CPA 3]			
1	2	3	4
The student is unfamiliar with the equipment and tools.	The student is aware of equipment and tools to some extent but can't use them adequately	The student is aware of tools and equipment but only partially confident in handling them	The student is proficient with the equipment usage.
Criterion 3: How well the student managed various issues during solution design? [CPA 1, CPA 2, CPA 3]			
1	2	3	4
No conflicting issues were managed or resolved at all.	Student could handle only some of the issues but not in a befitting manner.	Most of the issues were handled gracefully	All such issues were resolved with successful implementation.
Criterion 4: Has the student submitted all the deliverables on time? [CPA 1, CPA 2]			
1	2	3	4
Only one deliverable submitted	Only few deliverables are submitted.	All deliverables are submitted but not up to the mark and not on time.	All of the deliverables were submitted timely
			Total Marks:

Teacher's Signature: ____________

Design Methodology:

In our digital system design, we've harnessed FPGA technology to seamlessly integrate temperature sensor data. Leveraging I2C protocol for communication, the Temperature Sensor Interface Module (TSIM) efficiently extract data for temperature. The PWM Intensity Control Module (PWMIM) plays a pivotal role, generating Pulse Width Modulation signals to precisely manage the intensity of tri-color LEDs based on the received sensor data. Meanwhile, the LED Color Coding Module (LCCM) ensures an intuitive visual representation by mapping temperature sensor values to specific LED colors. The Display Module (DM) then takes charge, offering the flexibility to showcase sensor values on seven-segment display. Throughout the design process, we've emphasized modularity, simulation, FPGA implementation, testing, and iterative refinement to create a robust and adaptable system.

Design Decisions:

Temperature ranges:

Temperature (°C)	Intensity	Color
-30-0	High	White
1-5	Low	
6-10	Medium	
11-13	Low	Blue
14-16	Medium	
17-20	High	
21-23	Low	Yellow
24-26	Medium	
27-30	High	
Above 30	High	Red

PWM Duty Cycles:

Intensity	Duty Cycle
Low	90%
Medium	40%
High	10%

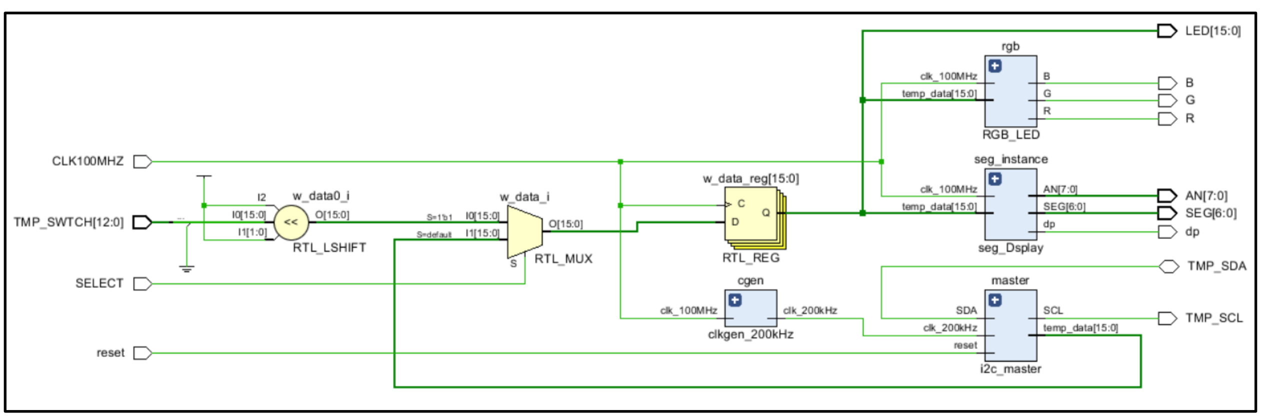
Verilog Code

```
//========================================================= RGB: START ==============
module RGB_LED(input clk_100MHz, input [15:0] temp_data, output reg R, G, B);

//5 bit pwm counter
wire [4:0] pwm_count;
//3 bit RGB_Color
reg [2:0] RGB_Color;
//reg [12:0] temperature;
reg signed [12:0] temperature;

always @(posedge clk_100MHz) begin
    if (temp_data[15]==1) //For negative temperature
        begin temperature <= (((temp_data[15:3])-8192)/16); end
    else                //For positive temperature
        begin temperature <= (temp_data[15:3])/16; end

//For Color Coding and variation in Intensity
if(-13'sd30 <= temperature && temperature <= 13'sd10) //temperature [-30,10]
begin
    RGB_Color <= 3'b111; // white
    if(-13'sd30 <= temperature && temperature <= 13'sd0) begin
        if (pwm_count < 27)  //high intensity, 90% duty cycle
        begin
            R <= RGB_Color[2]; G <= RGB_Color[1]; B <= RGB_Color[0];
        end
        else {R, G, B} <= 3'b000;
     end
     else if (13'sd5 < temperature && temperature <= 13'sd10) begin
        if (pwm_count < 12) //medium intensity, 40% duty cycle
        begin
            R <= RGB_Color[2]; G <= RGB_Color[1]; B <= RGB_Color[0];
        end
        else {R, G, B} <= 3'b000;
     end
     else if(13'sd0 < temperature && temperature <= 13'sd5) //temperature(0,5]
     begin
        if (pwm_count < 3)  //low intensity, 10% duty cycle
        begin
            R <= RGB_Color[2]; G <= RGB_Color[1]; B <= RGB_Color[0];
        end
        else {R, G, B} <= 3'b000;
     end
     else begin end
end
else if(10 < temperature && temperature <= 20) //temperature (10,20]
begin
    RGB_Color <= 3'b001; // blue
    if(10 < temperature && temperature <= 13) //temperature (10,13]
    begin
        if (pwm_count < 3) //low intensity, 10% duty cycle
        begin
            R <= RGB_Color[2]; G <= RGB_Color[1]; B <= RGB_Color[0];
        end
        else {R, G, B} <= 3'b000;
    end
    else if (13 < temperature && temperature <= 16) //temperature (13,16]
    begin
        if (pwm_count < 12) //medium intensity, 40% duty cycle
        begin
            R <= RGB_Color[2]; G <= RGB_Color[1]; B <= RGB_Color[0];
        end
        else {R, G, B} <= 3'b000;
    end
```

```
        else if(16 < temperature && temperature <= 20) //temperature (16,20]
        begin
            if (pwm_count < 27) //high intensity, 90% duty cycle
            begin
                R <= RGB_Color[2]; G <= RGB_Color[1]; B <= RGB_Color[0];
            end
            else {R, G, B} <= 3'b000;
        end
        else begin end
    end
    else if(20 < temperature && temperature <= 30) //temperature (20,30]
    begin
        RGB_Color <= 3'b110; // yellow
        if(20 < temperature && temperature <= 23) //temperature (20,23]
        begin
            if (pwm_count < 3) //low intensity, 10% duty cycle
            begin
                R <= RGB_Color[2]; G <= RGB_Color[1]; B <= RGB_Color[0];
            end
            else {R, G, B} <= 3'b000;
        end
        else if (23 < temperature && temperature <= 26) //temperature (23,26]
        begin
            if (pwm_count < 12) //medium intensity, 40% duty cycle
            begin
                R <= RGB_Color[2]; G <= RGB_Color[1]; B <= RGB_Color[0];
            end
            else {R, G, B} <= 3'b000;
        end
        else if (26 < temperature && temperature <= 30) //temperature (26,30]
        begin
            if (pwm_count < 27) //high intensity, 90% duty cycle
            begin
                R <= RGB_Color[2]; G <= RGB_Color[1]; B <= RGB_Color[0];
            end
            else {R, G, B} <= 3'b000;
        end
        else begin end
    end
    else if (30 < temperature) //temperature > 30
    begin
        RGB_Color <= 3'b100; // red
        if (pwm_count < 27) //High intensity, 90% duty cycle
        begin
            R <= RGB_Color[2]; G <= RGB_Color[1]; B <= RGB_Color[0];
        end
        else {R, G, B} <= 3'b000;
    end
    else {R, G, B} <= 3'b000;
    end

    //instance for pwm
    pwm p1(clk_100MHz,pwm_count);

    endmodule
    //========================================================== RGB: END ======
    //-----------------------------------------------PWM: START-----------------
    module pwm(input clk, output reg [3:0] pwm_count);

    reg [3:0] pwm_counter = 0;
    always @(posedge clk)
    begin
        if (pwm_counter < 16) pwm_count <= pwm_counter + 1; //count until 16
        else pwm_count <= 0; //reset counter
    end
    endmodule
    //-----------------------------------------------PWM: END-------------------
```

```
module temp_7seg_rgb_pwm(
    input           CLK100MHZ,          // nexys clk signal
    input           reset,              // btnC on nexys
    input [12:0]    TMP_SWTCH,          //Temperature value from Switches
    input           SELECT,             //To select temperature from sensor or switches SELECT=1,data from switches
    inout           TMP_SDA,            // i2c sda on temp sensor - bidirectional
    output          TMP_SCL,            // i2c scl on temp sensor
    output [6:0]    SEG,                // 7 segments of each display
    output [7:0]    AN,                 // 8 anodes of 8 displays
    output dp,                          // Decimal point of only 5th segment is ON
    output [15:0]   LED ,               // nexys leds = binary temp in deg C
    output R, G, B);                    // for RGB LED

    wire sda_dir;                       // direction of SDA signal - to or from master
    wire w_200kHz;                      // 200kHz SCL
    reg [15:0] w_data;                  // 8 bits of temperature data
    wire [15:0] temperature;            // 8 bits of temperature data from temperature sensor

    // Instantiate i2c master
    i2c_master master(.clk_200kHz(w_200kHz), .reset(reset), .temp_data(temperature),
        .SDA(TMP_SDA), .SDA_dir(sda_dir), .SCL(TMP_SCL));

     always@(posedge CLK100MHZ) begin
        if (SELECT==1) w_data<=(TMP_SWTCH<<3);
        else w_data<=temperature;
     end

    // Instantiate 200kHz clock generator
    clkgen_200kHz cgen(.clk_100MHz(CLK100MHZ), .clk_200kHz(w_200kHz));

    // Instantiate 7 segment control
    seg_Dsplay seg_instance(.clk_100MHz(CLK100MHZ), .temp_data(w_data), .SEG(SEG), .AN(AN), .dp(dp));
    // Set LED value to temp data
    assign LED = w_data;

    // Instantiate Tri-Color Led Color and Intensity Control
    RGB_LED rgb(.clk_100MHz(CLK100MHZ), .temp_data(w_data), .R(R), .G(G), .B(B));
endmodule
//=====================================================TOP MODULE: END=============================================
```

<u>RTL</u>



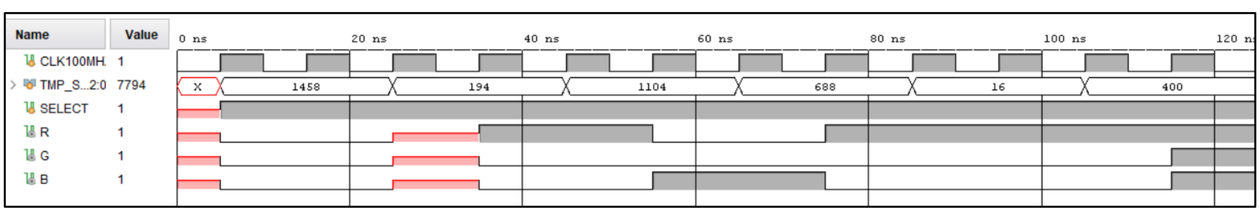
Constraint File

```
## Clock signal
set_property -dict { PACKAGE_PIN E3    IOSTANDARD LVCMOS33 } [get_ports { CLK100MHZ }]; #IO_L12P_T1_MRCC_35 Sch=clk100mhz
create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports {CLK100MHZ}];

## LEDs
set_property -dict { PACKAGE_PIN H17   IOSTANDARD LVCMOS33 } [get_ports { LED[0] }]; #IO_L18P_T2_A24_15 Sch=led[0]
set_property -dict { PACKAGE_PIN K15   IOSTANDARD LVCMOS33 } [get_ports { LED[1] }]; #IO_L24P_T3_RS1_15 Sch=led[1]
set_property -dict { PACKAGE_PIN J13   IOSTANDARD LVCMOS33 } [get_ports { LED[2] }]; #IO_L17N_T2_A25_15 Sch=led[2]
set_property -dict { PACKAGE_PIN N14   IOSTANDARD LVCMOS33 } [get_ports { LED[3] }]; #IO_L8P_T1_D11_14 Sch=led[3]
set_property -dict { PACKAGE_PIN R18   IOSTANDARD LVCMOS33 } [get_ports { LED[4] }]; #IO_L7P_T1_D09_14 Sch=led[4]
set_property -dict { PACKAGE_PIN V17   IOSTANDARD LVCMOS33 } [get_ports { LED[5] }]; #IO_L18N_T2_A11_D27_14 Sch=led[5]
set_property -dict { PACKAGE_PIN U17   IOSTANDARD LVCMOS33 } [get_ports { LED[6] }]; #IO_L17P_T2_A14_D30_14 Sch=led[6]
set_property -dict { PACKAGE_PIN U16   IOSTANDARD LVCMOS33 } [get_ports { LED[7] }]; #IO_L18P_T2_A12_D28_14 Sch=led[7]
set_property -dict { PACKAGE_PIN V16   IOSTANDARD LVCMOS33 } [get_ports { LED[8] }]; #IO_L16N_T2_A15_D31_14 Sch=led[8]
set_property -dict { PACKAGE_PIN T15   IOSTANDARD LVCMOS33 } [get_ports { LED[9] }]; #IO_L14N_T2_SRCC_14 Sch=led[9]
set_property -dict { PACKAGE_PIN U14   IOSTANDARD LVCMOS33 } [get_ports { LED[10] }]; #IO_L22P_T3_A05_D21_14 Sch=led[10]
set_property -dict { PACKAGE_PIN T16   IOSTANDARD LVCMOS33 } [get_ports { LED[11] }]; #IO_L15N_T2_DQS_DOUT_CSO_B_14 Sch=led[11]
set_property -dict { PACKAGE_PIN V15   IOSTANDARD LVCMOS33 } [get_ports { LED[12] }]; #IO_L16P_T2_CSI_B_14 Sch=led[12]
set_property -dict { PACKAGE_PIN V14   IOSTANDARD LVCMOS33 } [get_ports { LED[13] }]; #IO_L22N_T3_A04_D20_14 Sch=led[13]
set_property -dict { PACKAGE_PIN V12   IOSTANDARD LVCMOS33 } [get_ports { LED[14] }]; #IO_L20N_T3_A07_D23_14 Sch=led[14]
set_property -dict { PACKAGE_PIN V11   IOSTANDARD LVCMOS33 } [get_ports { LED[15] }]; #IO_L21N_T3_DQS_A06_D22_14 Sch=led[15]

##Buttons
set_property -dict { PACKAGE_PIN N17   IOSTANDARD LVCMOS33 } [get_ports { reset }]; #IO_L9P_T1_DQS_14 Sch=btnc

##7 segment display
set_property -dict { PACKAGE_PIN T10   IOSTANDARD LVCMOS33 } [get_ports { SEG[0] }]; #IO_L24N_T3_A00_D16_14 Sch=ca
set_property -dict { PACKAGE_PIN R10   IOSTANDARD LVCMOS33 } [get_ports { SEG[1] }]; #IO_25_14 Sch=cb
set_property -dict { PACKAGE_PIN K16   IOSTANDARD LVCMOS33 } [get_ports { SEG[2] }]; #IO_25_15 Sch=cc
set_property -dict { PACKAGE_PIN K13   IOSTANDARD LVCMOS33 } [get_ports { SEG[3] }]; #IO_L17P_T2_A26_15 Sch=cd
set_property -dict { PACKAGE_PIN P15   IOSTANDARD LVCMOS33 } [get_ports { SEG[4] }]; #IO_L13P_T2_MRCC_14 Sch=ce
set_property -dict { PACKAGE_PIN T11   IOSTANDARD LVCMOS33 } [get_ports { SEG[5] }]; #IO_L19P_T3_A10_D26_14 Sch=cf
set_property -dict { PACKAGE_PIN L18   IOSTANDARD LVCMOS33 } [get_ports { SEG[6] }]; #IO_L4P_T0_D04_14 Sch=cg
set_property -dict { PACKAGE_PIN H15   IOSTANDARD LVCMOS33 } [get_ports { dp }]; #IO_L19N_T3_A21_VREF_15 Sch=dp
set_property -dict { PACKAGE_PIN J17   IOSTANDARD LVCMOS33 } [get_ports { AN[0] }]; #IO_L23P_T3_FOE_B_15 Sch=an[0]
set_property -dict { PACKAGE_PIN J18   IOSTANDARD LVCMOS33 } [get_ports { AN[1] }]; #IO_L23N_T3_FWE_B_15 Sch=an[1]
set_property -dict { PACKAGE_PIN T9    IOSTANDARD LVCMOS33 } [get_ports { AN[2] }]; #IO_L24P_T3_A01_D17_14 Sch=an[2]
set_property -dict { PACKAGE_PIN J14   IOSTANDARD LVCMOS33 } [get_ports { AN[3] }]; #IO_L19P_T3_A22_15 Sch=an[3]
set_property -dict { PACKAGE_PIN P14   IOSTANDARD LVCMOS33 } [get_ports { AN[4] }]; #IO_L8N_T1_D12_14 Sch=an[4]
set_property -dict { PACKAGE_PIN T14   IOSTANDARD LVCMOS33 } [get_ports { AN[5] }]; #IO_L14P_T2_SRCC_14 Sch=an[5]
set_property -dict { PACKAGE_PIN K2    IOSTANDARD LVCMOS33 } [get_ports { AN[6] }]; #IO_L23P_T3_35 Sch=an[6]
set_property -dict { PACKAGE_PIN U13   IOSTANDARD LVCMOS33 } [get_ports { AN[7] }]; #IO_L23N_T3_A02_D18_14 Sch=an[7]

## RGB LEDs
set_property -dict { PACKAGE_PIN R12   IOSTANDARD LVCMOS33 } [get_ports { B }]; #IO_L5P_T0_D06_14 Sch=led16_b
set_property -dict { PACKAGE_PIN M16   IOSTANDARD LVCMOS33 } [get_ports { G }]; #IO_L10P_T1_D14_14 Sch=led16_g
set_property -dict { PACKAGE_PIN N15   IOSTANDARD LVCMOS33 } [get_ports { R }]; #IO_L11P_T1_SRCC_14 Sch=led16_r

##Temperature Sensor
set_property -dict { PACKAGE_PIN C14   IOSTANDARD LVCMOS33 } [get_ports { TMP_SCL }]; #IO_L1N_T0_AD0N_15 Sch=tmp_scl
set_property -dict { PACKAGE_PIN C15   IOSTANDARD LVCMOS33 } [get_ports { TMP_SDA }]; #IO_L12N_T1_MRCC_15 Sch=tmp_sda
#set_property -dict { PACKAGE_PIN D13   IOSTANDARD LVCMOS33 } [get_ports { TMP_INT }]; #IO_L6N_T0_VREF_15 Sch=tmp_int
#set_property -dict { PACKAGE_PIN B14   IOSTANDARD LVCMOS33 } [get_ports { TMP_CT }]; #IO_L2N_T0_AD8N_15 Sch=tmp_ct

##Switches
set_property -dict { PACKAGE_PIN J15 IOSTANDARD LVCMOS33 } [get_ports { TMP_SWTCH[0] }];
set_property -dict { PACKAGE_PIN L16 IOSTANDARD LVCMOS33 } [get_ports { TMP_SWTCH[1] }];
set_property -dict { PACKAGE_PIN M13 IOSTANDARD LVCMOS33 } [get_ports { TMP_SWTCH[2] }];
set_property -dict { PACKAGE_PIN R15 IOSTANDARD LVCMOS33 } [get_ports { TMP_SWTCH[3] }];
set_property -dict { PACKAGE_PIN R17 IOSTANDARD LVCMOS33 } [get_ports { TMP_SWTCH[4] }];
set_property -dict { PACKAGE_PIN T18 IOSTANDARD LVCMOS33 } [get_ports { TMP_SWTCH[5] }];
set_property -dict { PACKAGE_PIN U18 IOSTANDARD LVCMOS33 } [get_ports { TMP_SWTCH[6] }];
set_property -dict { PACKAGE_PIN R13 IOSTANDARD LVCMOS33 } [get_ports { TMP_SWTCH[7] }];
set_property -dict { PACKAGE_PIN T8 IOSTANDARD LVCMOS33 } [get_ports { TMP_SWTCH[8] }];
set_property -dict { PACKAGE_PIN U8 IOSTANDARD LVCMOS33 } [get_ports { TMP_SWTCH[9] }];
set_property -dict { PACKAGE_PIN R16 IOSTANDARD LVCMOS33 } [get_ports { TMP_SWTCH[10] }];
set_property -dict { PACKAGE_PIN T13 IOSTANDARD LVCMOS33 } [get_ports { TMP_SWTCH[11] }];
set_property -dict { PACKAGE_PIN H6 IOSTANDARD LVCMOS33 } [get_ports { TMP_SWTCH[12] }];

##One switch for selecting temperature input from sensor or from switches
set_property -dict { PACKAGE_PIN V10 IOSTANDARD LVCMOS33 } [get_ports { SELECT }];
```

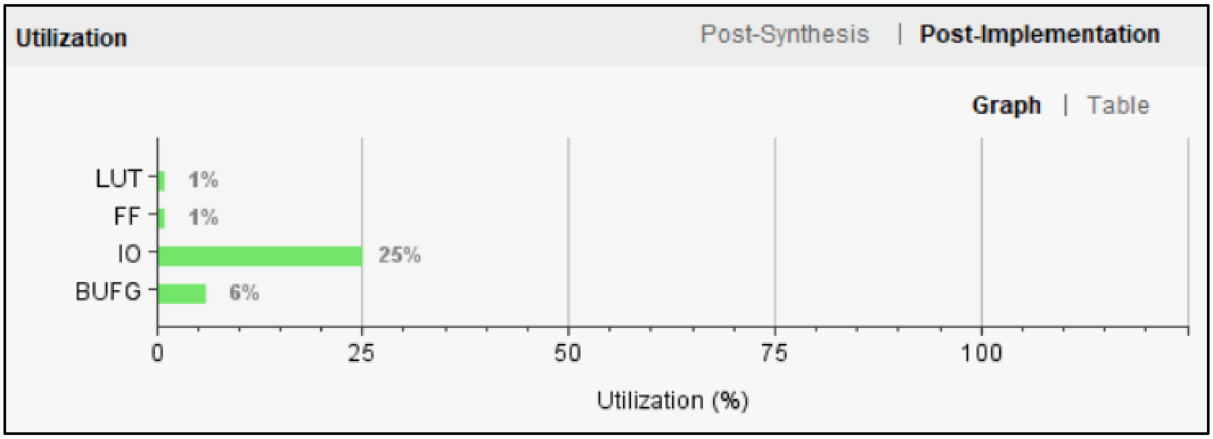
Testbench

```
module temp_7seg_rgb_pwm_TB;
    reg          CLK100MHZ;       // nexys clk signal
    reg  [12:0]  TMP_SWTCH;       //Temperature value from Switches
    reg          SELECT;          //To select temperature from sensor or switches SELECT=1,data from switches
    wire R;                       //for TRI-COLOR LED
    wire G;                       //for TRI-COLOR LED
    wire B;                       //for TRI-COLOR LED


 temp_7seg_rgb_pwm uut(.CLK100MHZ(CLK100MHZ),
    .TMP_SWTCH(TMP_SWTCH),
    .SELECT(SELECT),
    .R(R),
    .G(G),
    .B(B)
   );


always #5 CLK100MHZ = ~CLK100MHZ;

initial begin

CLK100MHZ = 0;

//reading from temperature sensor
//SELECT = 0;

//#50;
//reading from switch
#5;
SELECT =1;
TMP_SWTCH = 13'b0010110110010;

#20;
TMP_SWTCH = 13'b0000011000010;

#20;
TMP_SWTCH =13'sb0010001010000;
```

Simulation



Resource Utilization

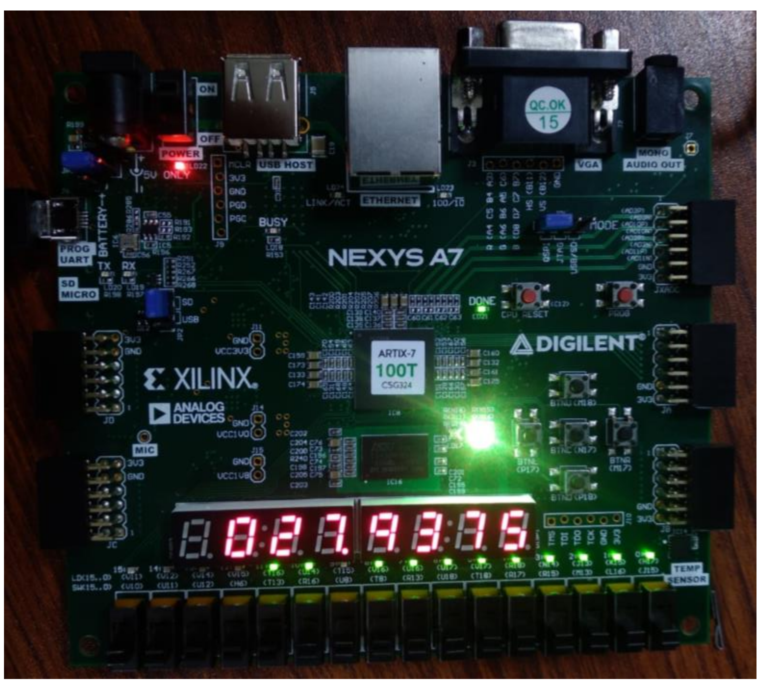


Utilization   Post-Synthesis | **Post-Implementation**

Graph | **Table**

Resource	Utilization	Available	Utilization %
LUT	333	63400	0.53
FF	113	126800	0.09
IO	53	210	25.24
BUFG	2	32	6.25

Physical Implementation



Applications

- Temperature sensors regulate indoor climate in HVAC systems.
- Temperature sensors prevent spoilage in food processing and storage.
- In healthcare, temperature sensors monitor and regulate body temperature.
- Industrial temperature sensors ensure optimal efficiency in manufacturing.
- Automotive temperature sensors prevent engine overheating.
- Temperature sensors contribute to weather forecasting in weather stations.
- In smart homes, temperature sensors optimize energy consumption.
- Electronic devices use temperature sensors to prevent overheating.
- Agriculture employs temperature sensors for optimal planting times.
- Scientific research relies on temperature sensors for accurate experiments.