SMART LIGHT – PROJECT REPORT

TEAM

ABHISHEK SAM
2019A3PS0363H

ARJUN REDDY SOMI REDDY
2019AAPS0301H

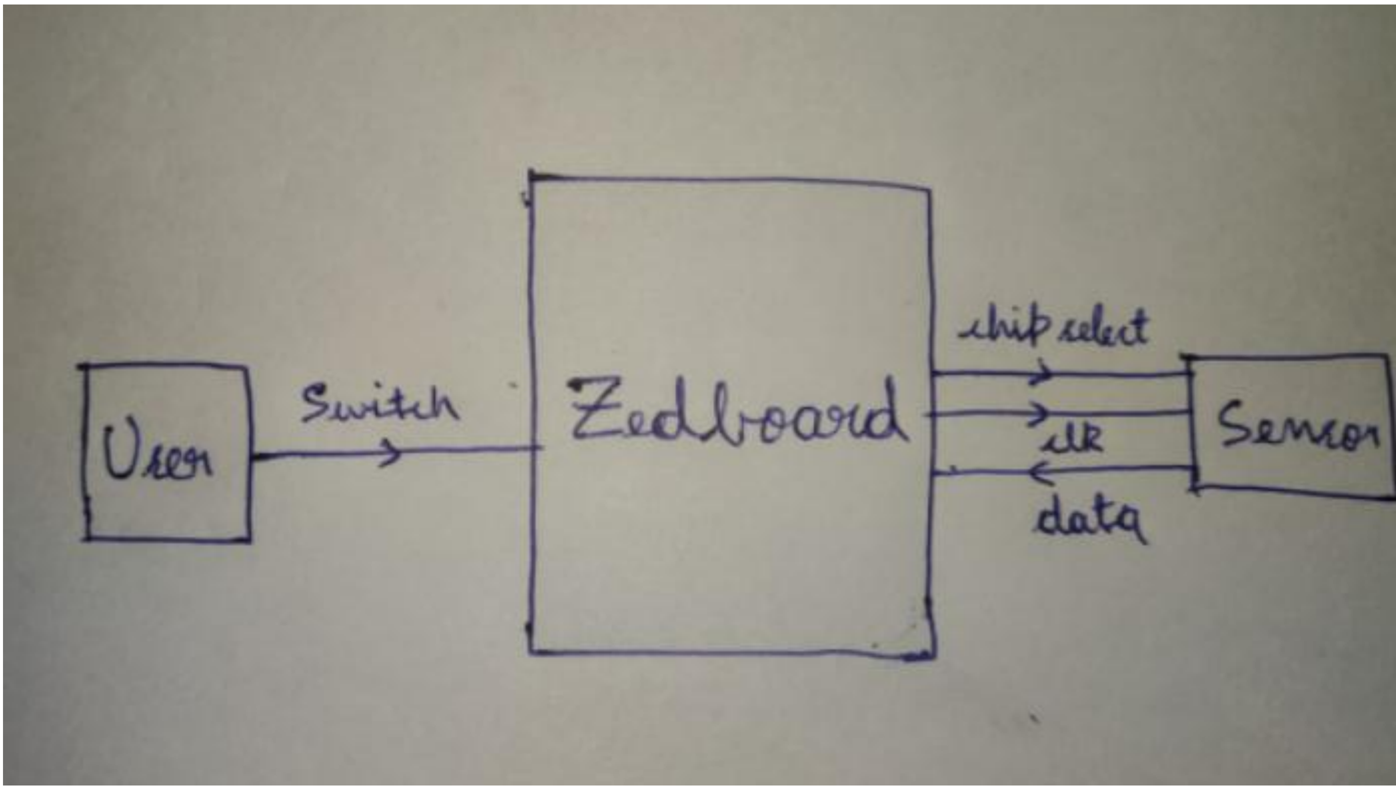
MANIKANTA SOMAYAJI
2019A3PS1339H
YARRAMSETTY SANJEEVA SAI PREETHAM
2019A3PS0485H

MOTIVATION:

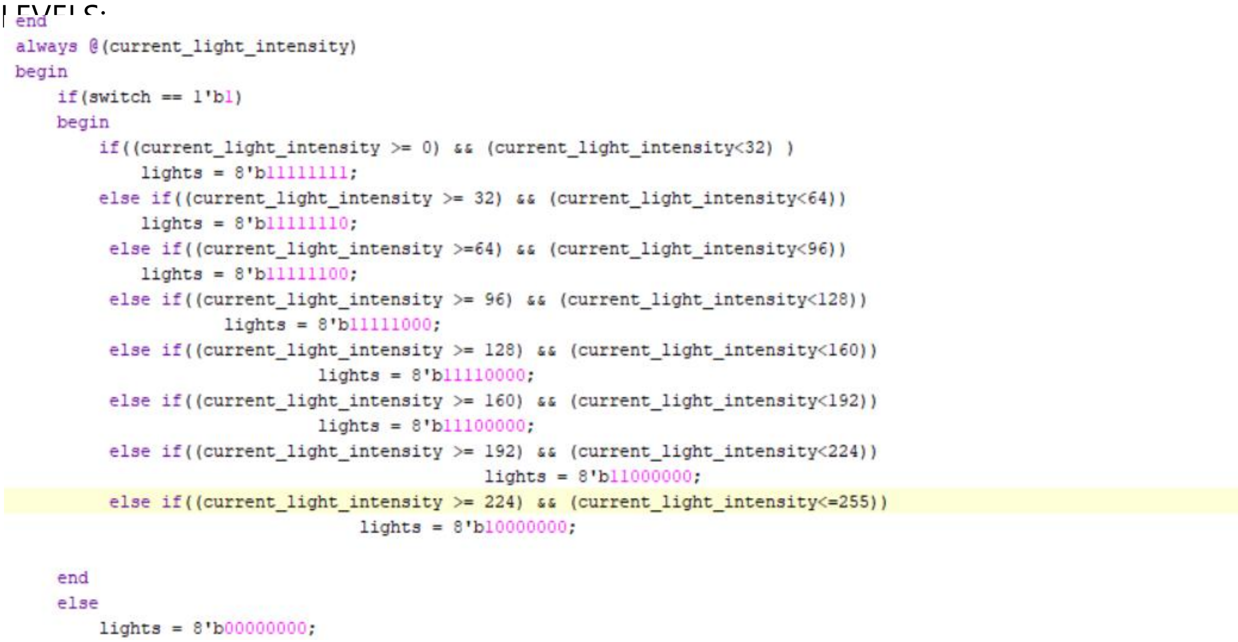
While world is moving to towards energy conservation and power saving, it proven that small changes in daily routines of people towards energy conservation, can collectively make a big difference in energy consumption. And this project is an attempt to bring such small change by effectively using natural light.

DESCRIPTION:

When we turn ON the switch, depending on the availability of natural light inside the room, number of LEDs turning on varies and when the switch if OFF, irrespective of amount of natural light, the light will be OFF. This is realised by using an Ambient Light Sensor (ALS) together with Zed Board. The DIP switch on Zed Board plays the role of the switch. When it is ON, Zed Board collects data (amount of natural light) from sensor and compare it with threshold values to get number of LEDs which should be ON. If switch is ON depending upon the natural light number of LEDs form 1-8 gets on and when the switch is OFF, the led is OFF.


User
Switch
Zedboard
chip select
clk
data
Sensor

LEVELS:

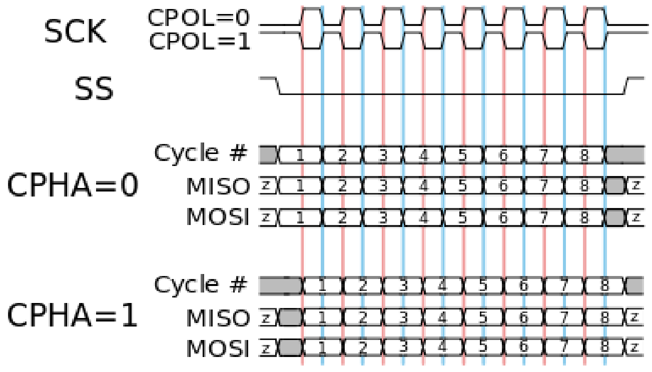


```
end
always @(current_light_intensity)
begin
    if(switch == 1'b1)
    begin
        if((current_light_intensity >= 0) && (current_light_intensity<32) )
            lights = 8'b11111111;
        else if((current_light_intensity >= 32) && (current_light_intensity<64))
            lights = 8'b11111110;
         else if((current_light_intensity >=64) && (current_light_intensity<96))
            lights = 8'b11111100;
         else if((current_light_intensity >= 96) && (current_light_intensity<128))
                    lights = 8'b11111000;
         else if((current_light_intensity >= 128) && (current_light_intensity<160))
                             lights = 8'b11110000;
         else if((current_light_intensity >= 160) && (current_light_intensity<192))
                             lights = 8'b11100000;
         else if((current_light_intensity >= 192) && (current_light_intensity<224))
                                             lights = 8'b11000000;
         else if((current_light_intensity >= 224) && (current_light_intensity<=255))
                                 lights = 8'b10000000;

    end
    else
        lights = 8'b00000000;
```

AMBIENT LIGHT SENSOR INTERFACING:

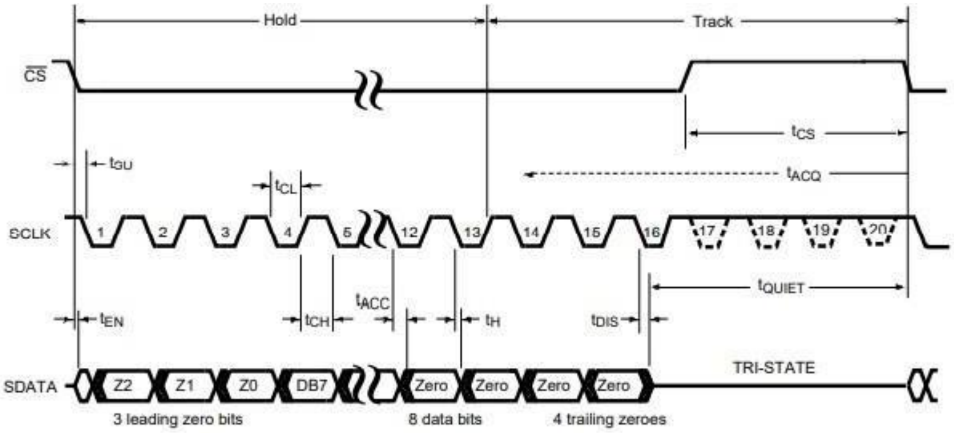
The biggest challenge while doing this project is interfacing ALS to Zed Board. As we need to implement SPI protocol on Zed-Board to get the data from ALS.

SPI Protocol (Serial Peripheral Interface) is a synchronous serial communication interface specification used for short-distance communication, primarily in embedded systems.



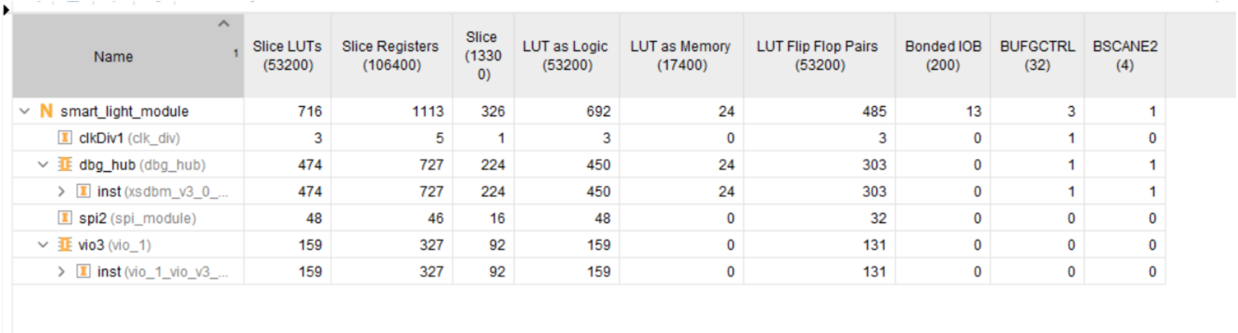
PmodALS which was used in this project uses ADC081S021 from Texas instruments as ADC converter. From data sheet, it became clear that it is SPI mode 3 i.e, CPOL = 1, CPHA = 1. It means that leading edge of SCLK is falling edge (clock polarity) and data is changed on every leading edge (clock phase).

When ~CS is LOW, the sensor needs 16 clock cycles to send 15bits of data where first 3 bits and last 4 bits are leading and trailing zeros respectively. The exact timing of ADC081S021 is shown below.

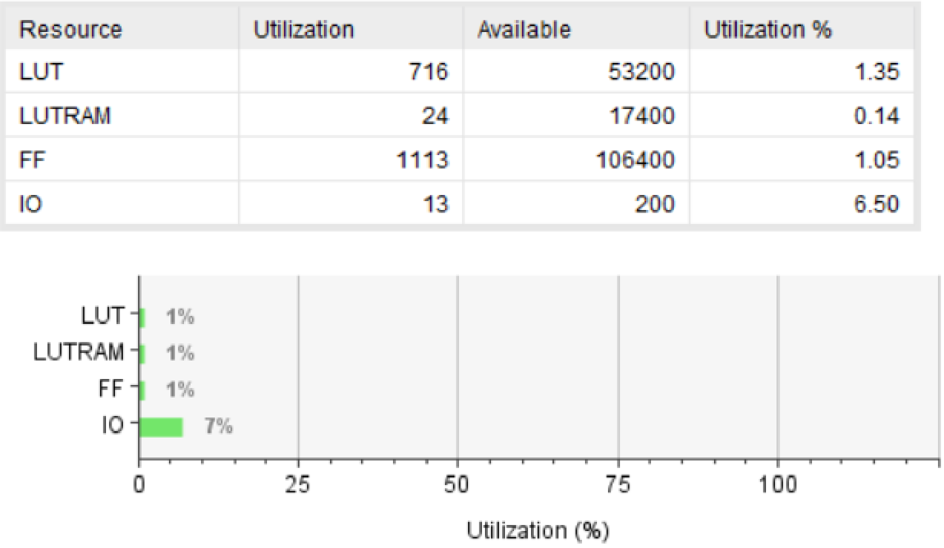


POST IMPLEMENTATION RESULTS:

- **LUTs**

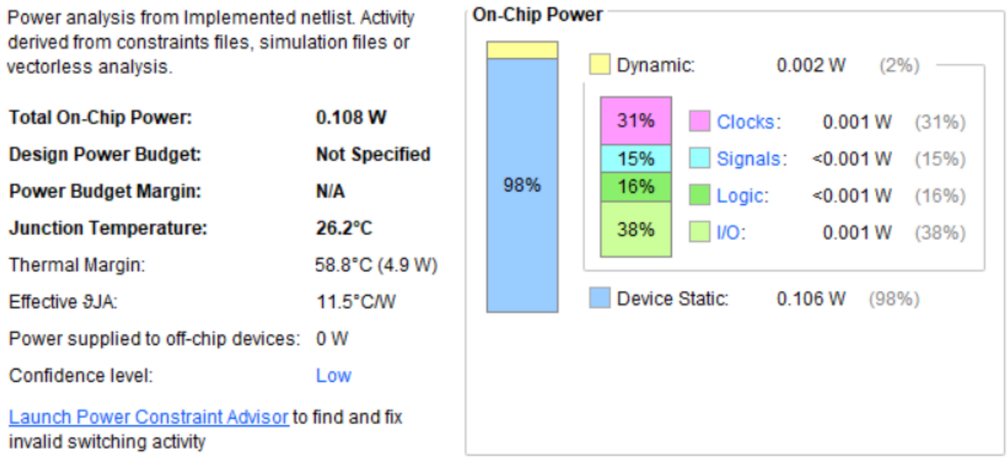


Name	Slice LUTs (53200)	Slice Registers (106400)	Slice (13300)	LUT as Logic (53200)	LUT as Memory (17400)	LUT Flip Flop Pairs (53200)	Bonded IOB (200)	BUFGCTRL (32)	BSCANE2 (4)
smart_light_module	716	1113	326	692	24	485	13	3	1
clkDiv1 (clk_div)	3	5	1	3	0	3	0	1	0
dbg_hub (dbg_hub)	474	727	224	450	24	303	0	1	1
inst (xsdbm_v3_0_...	474	727	224	450	24	303	0	1	1
spi2 (spi_module)	48	46	16	48	0	32	0	0	0
vio3 (vio_1)	159	327	92	159	0	131	0	0	0
inst (vio_1_vio_v3_...	159	327	92	159	0	131	0	0	0



Resource	Utilization	Available	Utilization %
LUT	716	53200	1.35
LUTRAM	24	17400	0.14
FF	1113	106400	1.05
IO	13	200	6.50

- **POWER**



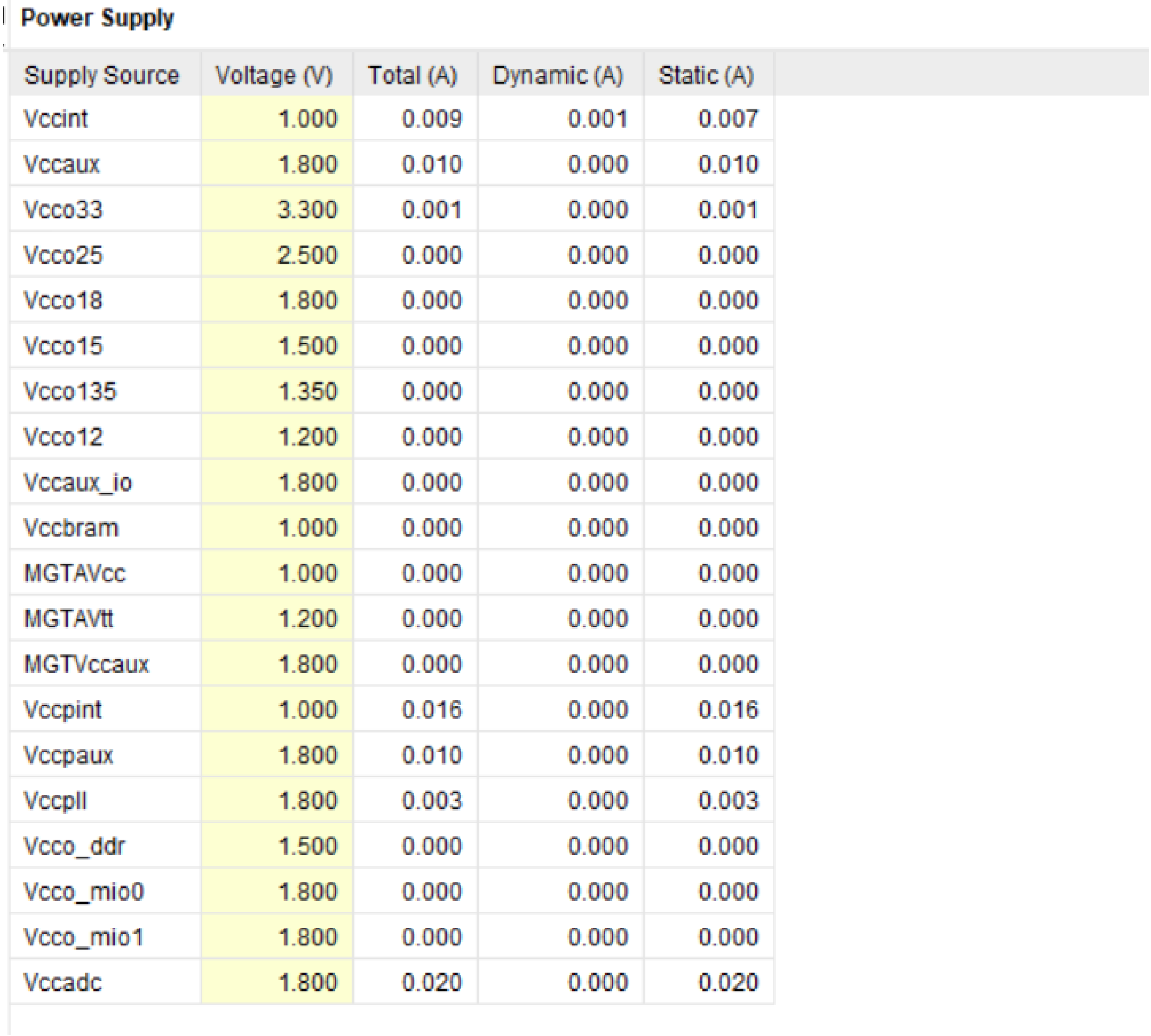
Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

Total On-Chip Power:	**0.108 W**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**26.2°C**
Thermal Margin:	58.8°C (4.9 W)
Effective ϑJA:	11.5°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Launch Power Constraint Advisor to find and fix invalid switching activity

On-Chip Power

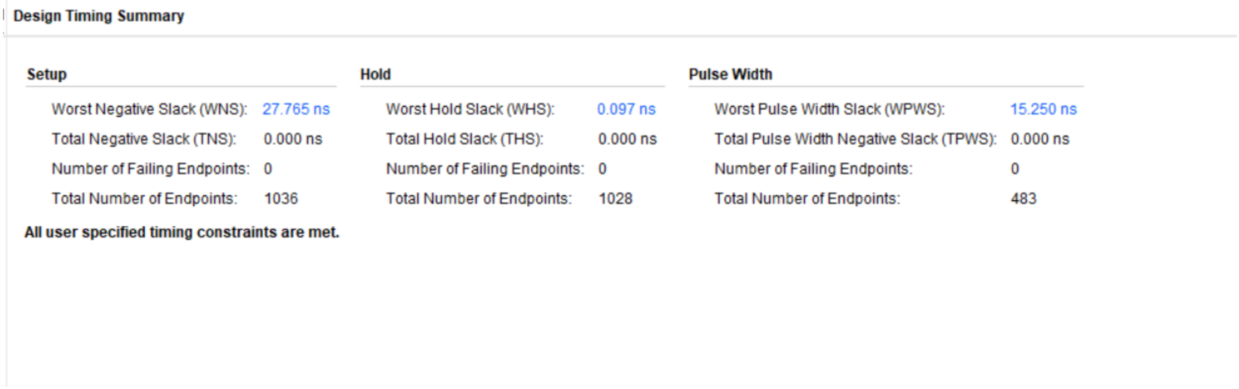
Dynamic:	0.002 W	(2%)
Clocks:	0.001 W	(31%)
Signals:	<0.001 W	(15%)
Logic:	<0.001 W	(16%)
I/O:	0.001 W	(38%)
Device Static:	0.106 W	(98%)



Power Supply

Supply Source	Voltage (V)	Total (A)	Dynamic (A)	Static (A)
Vccint	1.000	0.009	0.001	0.007
Vccaux	1.800	0.010	0.000	0.010
Vcco33	3.300	0.001	0.000	0.001
Vcco25	2.500	0.000	0.000	0.000
Vcco18	1.800	0.000	0.000	0.000
Vcco15	1.500	0.000	0.000	0.000
Vcco135	1.350	0.000	0.000	0.000
Vcco12	1.200	0.000	0.000	0.000
Vccaux_io	1.800	0.000	0.000	0.000
Vccbram	1.000	0.000	0.000	0.000
MGTAVcc	1.000	0.000	0.000	0.000
MGTAVtt	1.200	0.000	0.000	0.000
MGTVccaux	1.800	0.000	0.000	0.000
Vccpint	1.000	0.016	0.000	0.016
Vccpaux	1.800	0.010	0.000	0.010
Vccpll	1.800	0.003	0.000	0.003
Vcco_ddr	1.500	0.000	0.000	0.000
Vcco_mio0	1.800	0.000	0.000	0.000
Vcco_mio1	1.800	0.000	0.000	0.000
Vccadc	1.800	0.020	0.000	0.020

TIMING SUMMARY

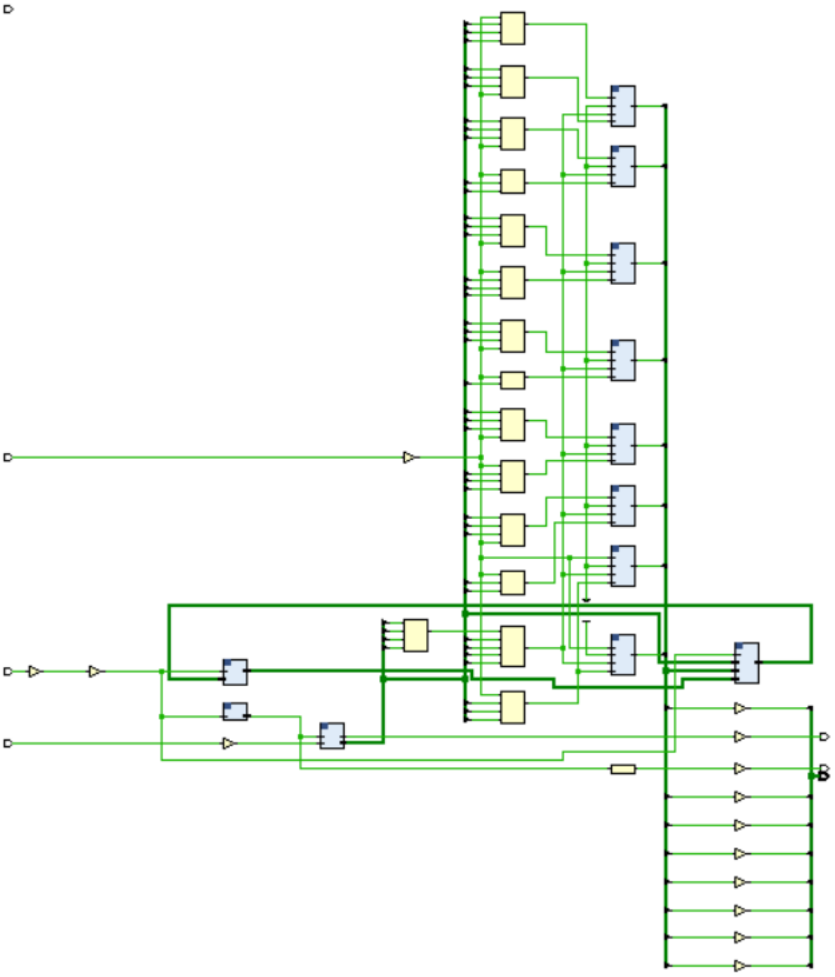


Design Timing Summary

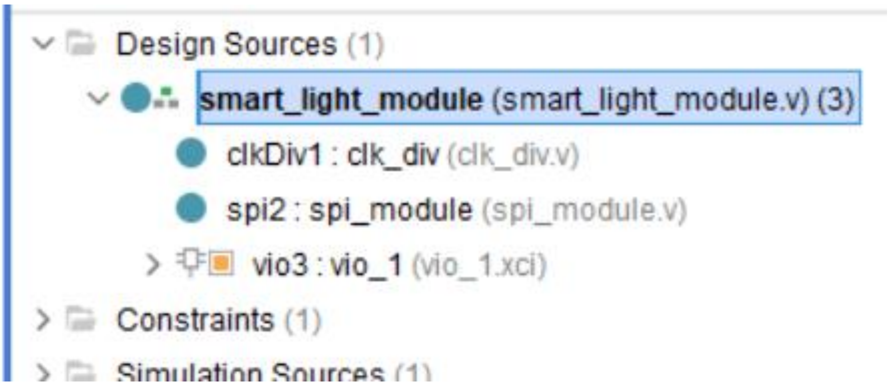
Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	27.765 ns	Worst Hold Slack (WHS):	0.097 ns	Worst Pulse Width Slack (WPWS):	15.250 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	1036	Total Number of Endpoints:	1028	Total Number of Endpoints:	483

All user specified timing constraints are met.

schematic

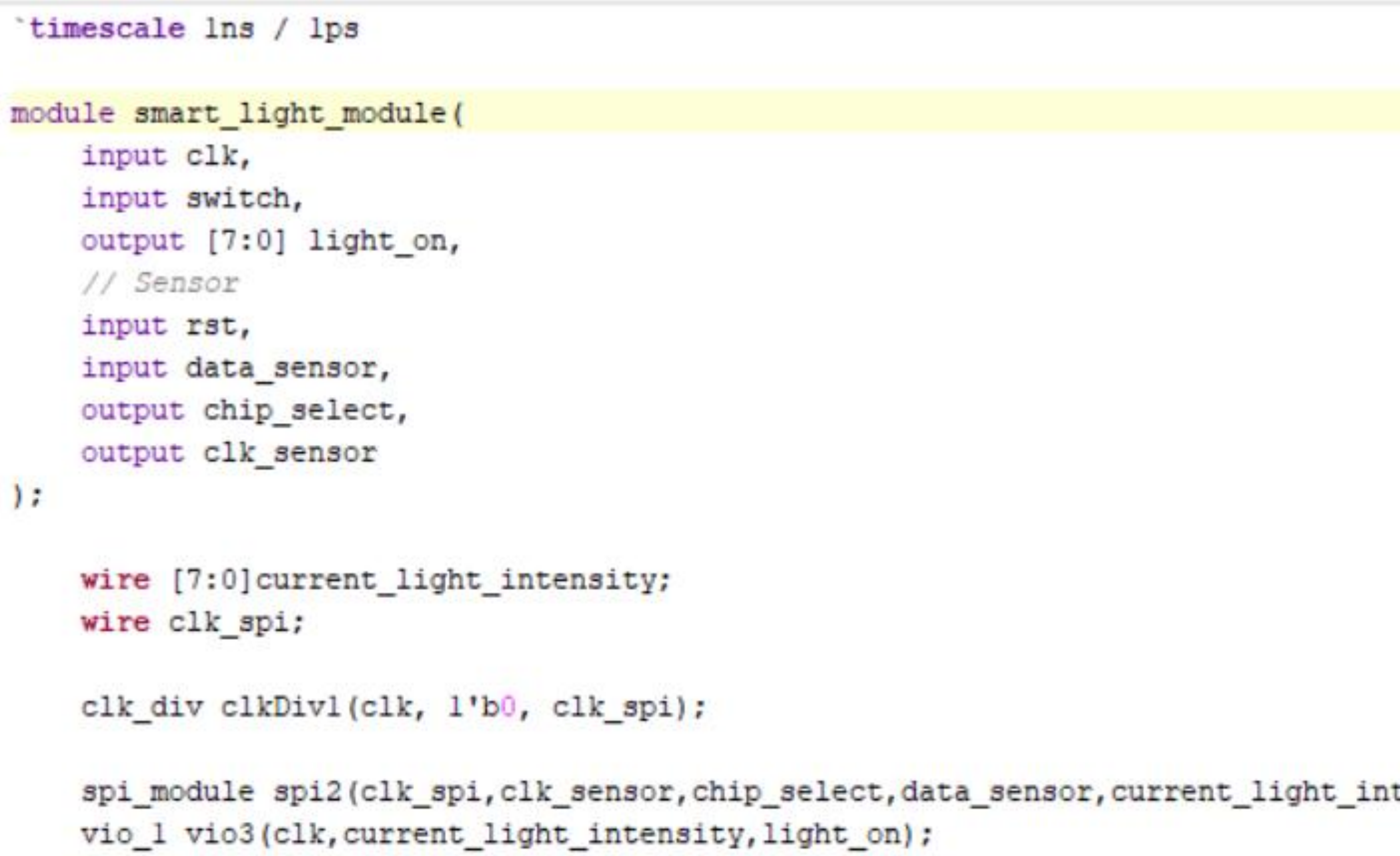


Hierarchial structure in code

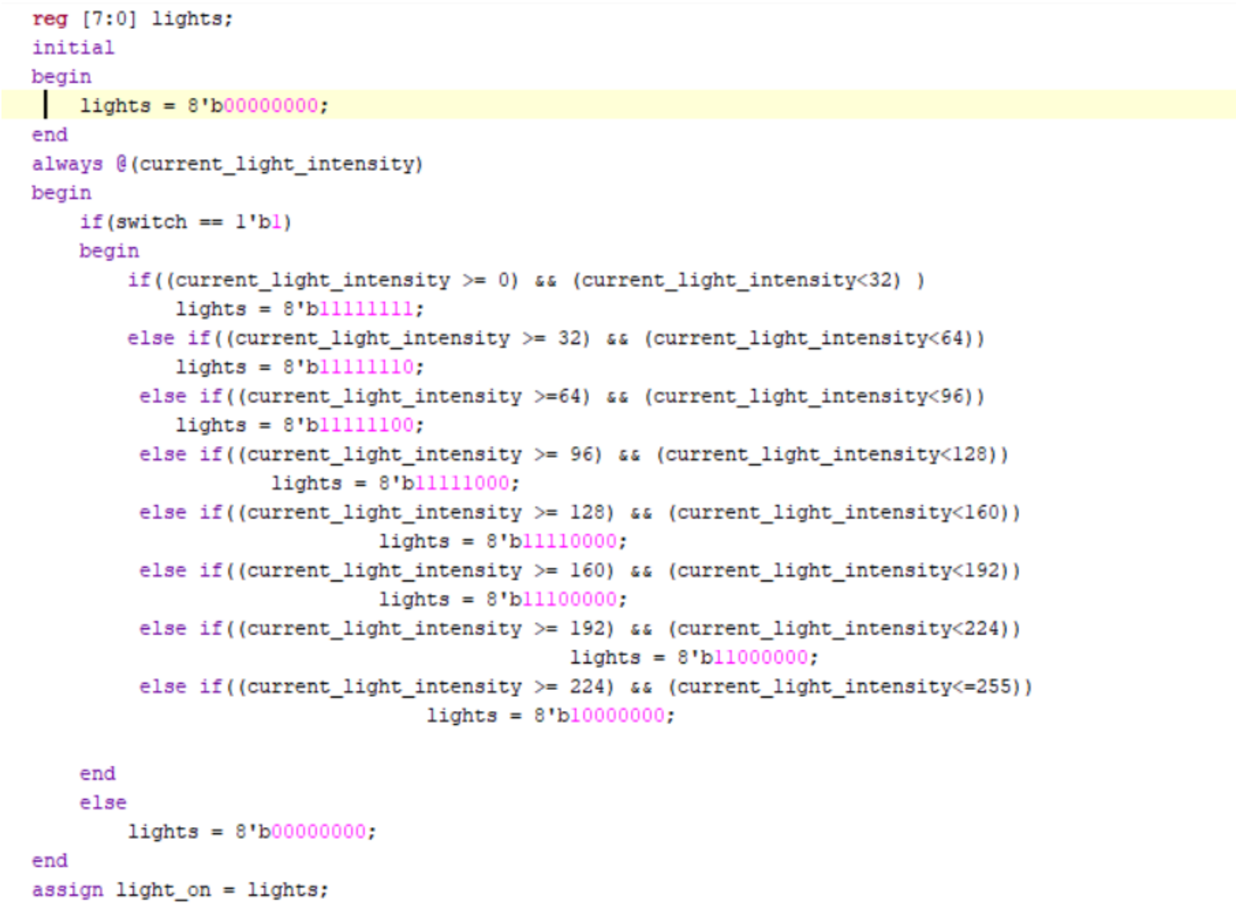


```
Design Sources (1)
    smart_light_module (smart_light_module.v) (3)
        clkDiv1 : clk_div (clk_div.v)
        spi2 : spi_module (spi_module.v)
        vio3 : vio_1 (vio_1.xci)
Constraints (1)
Simulation Sources (1)
```

Main:

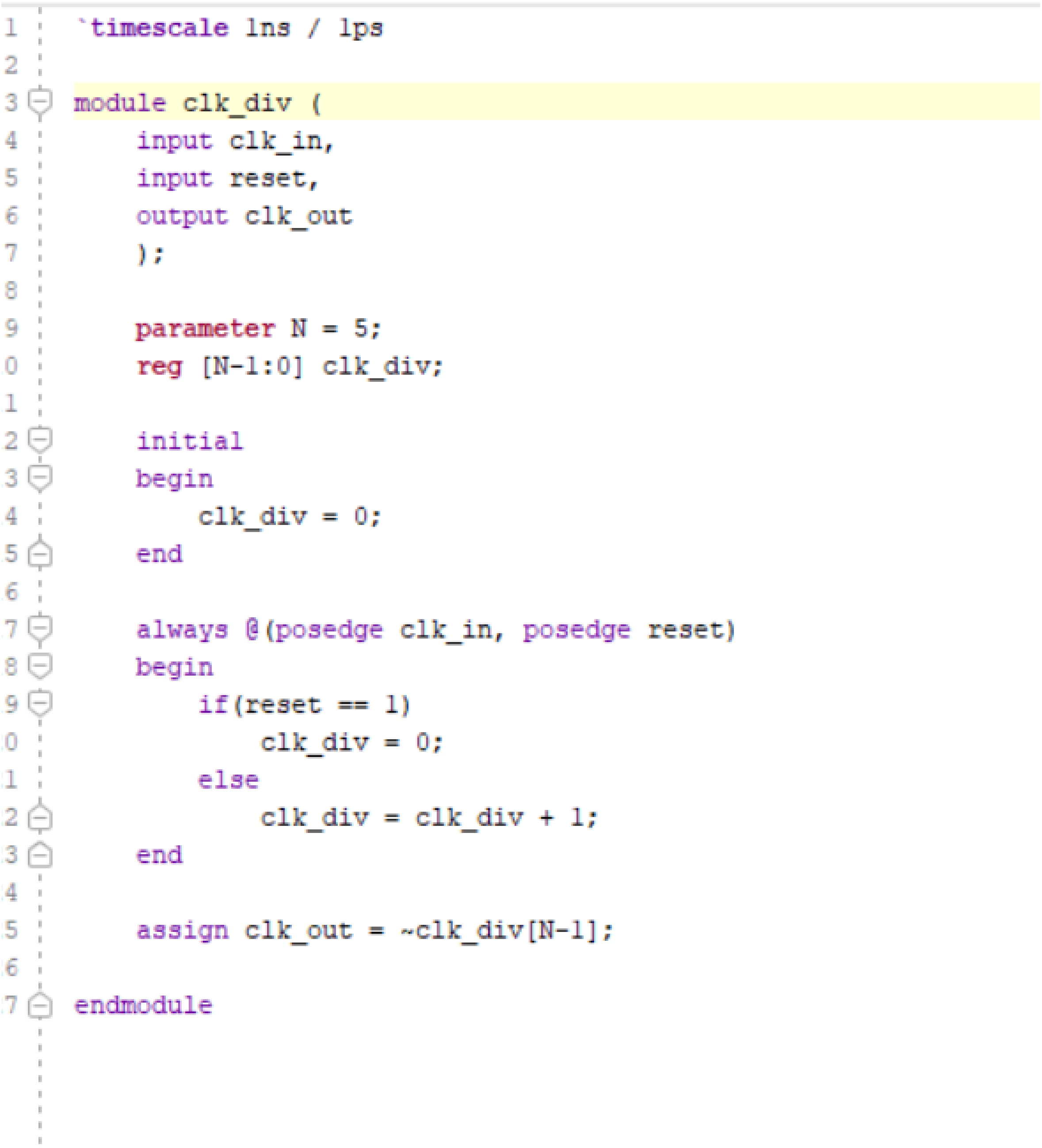


```
`timescale 1ns / 1ps

module smart_light_module(
    input clk,
    input switch,
    output [7:0] light_on,
    // Sensor
    input rst,
    input data_sensor,
    output chip_select,
    output clk_sensor
);

    wire [7:0]current_light_intensity;
    wire clk_spi;

    clk_div clkDiv1(clk, 1'b0, clk_spi);

    spi_module spi2(clk_spi,clk_sensor,chip_select,data_sensor,current_light_int
    vio_1 vio3(clk,current_light_intensity,light_on);
```



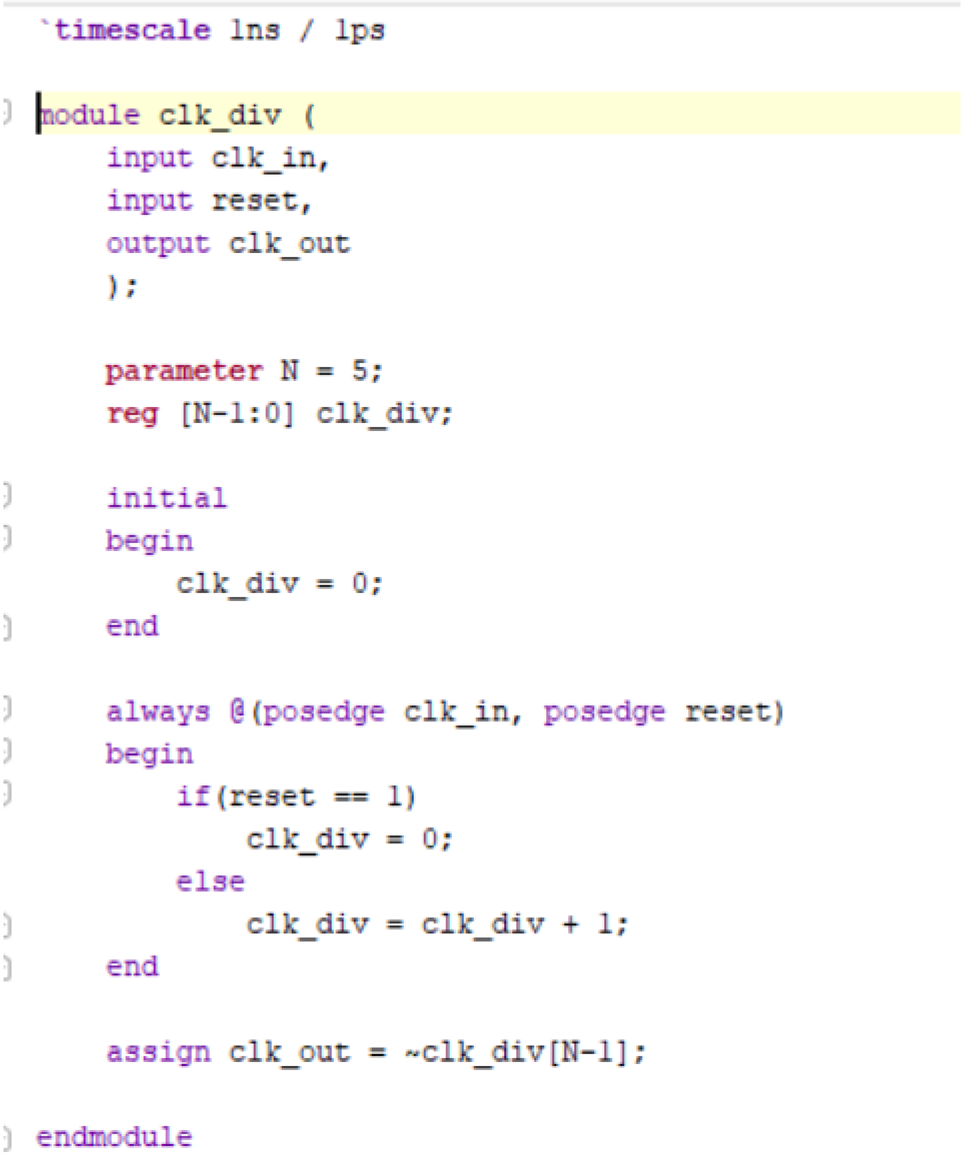
```verilog
reg [7:0] lights;
initial
begin
    lights = 8'b00000000;
end
always @(current_light_intensity)
begin
    if(switch == 1'b1)
    begin
        if((current_light_intensity >= 0) && (current_light_intensity<32) )
            lights = 8'b11111111;
        else if((current_light_intensity >= 32) && (current_light_intensity<64))
            lights = 8'b11111110;
         else if((current_light_intensity >=64) && (current_light_intensity<96))
            lights = 8'b11111100;
         else if((current_light_intensity >= 96) && (current_light_intensity<128))
                    lights = 8'b11111000;
         else if((current_light_intensity >= 128) && (current_light_intensity<160))
                             lights = 8'b11110000;
         else if((current_light_intensity >= 160) && (current_light_intensity<192))
                             lights = 8'b11100000;
         else if((current_light_intensity >= 192) && (current_light_intensity<224))
                                             lights = 8'b11000000;
         else if((current_light_intensity >= 224) && (current_light_intensity<=255))
                                 lights = 8'b10000000;

    end
    else
        lights = 8'b00000000;
end
assign light_on = lights;
```

This is the main module it calls necessary module like clk division and spi module and in this we are also deciding how many LEDS should be ON
we are using vio to display the leds which are glowing and the light intensity value received from pmod ALS



```verilog
`timescale 1ns / 1ps

module clk_div (
    input clk_in,
    input reset,
    output clk_out
    );

    parameter N = 5;
    reg [N-1:0] clk_div;

    initial
    begin
        clk_div = 0;
    end

    always @(posedge clk_in, posedge reset)
    begin
        if(reset == 1)
            clk_div = 0;
        else
            clk_div = clk_div + 1;
    end

    assign clk_out = ~clk_div[N-1];

endmodule
```

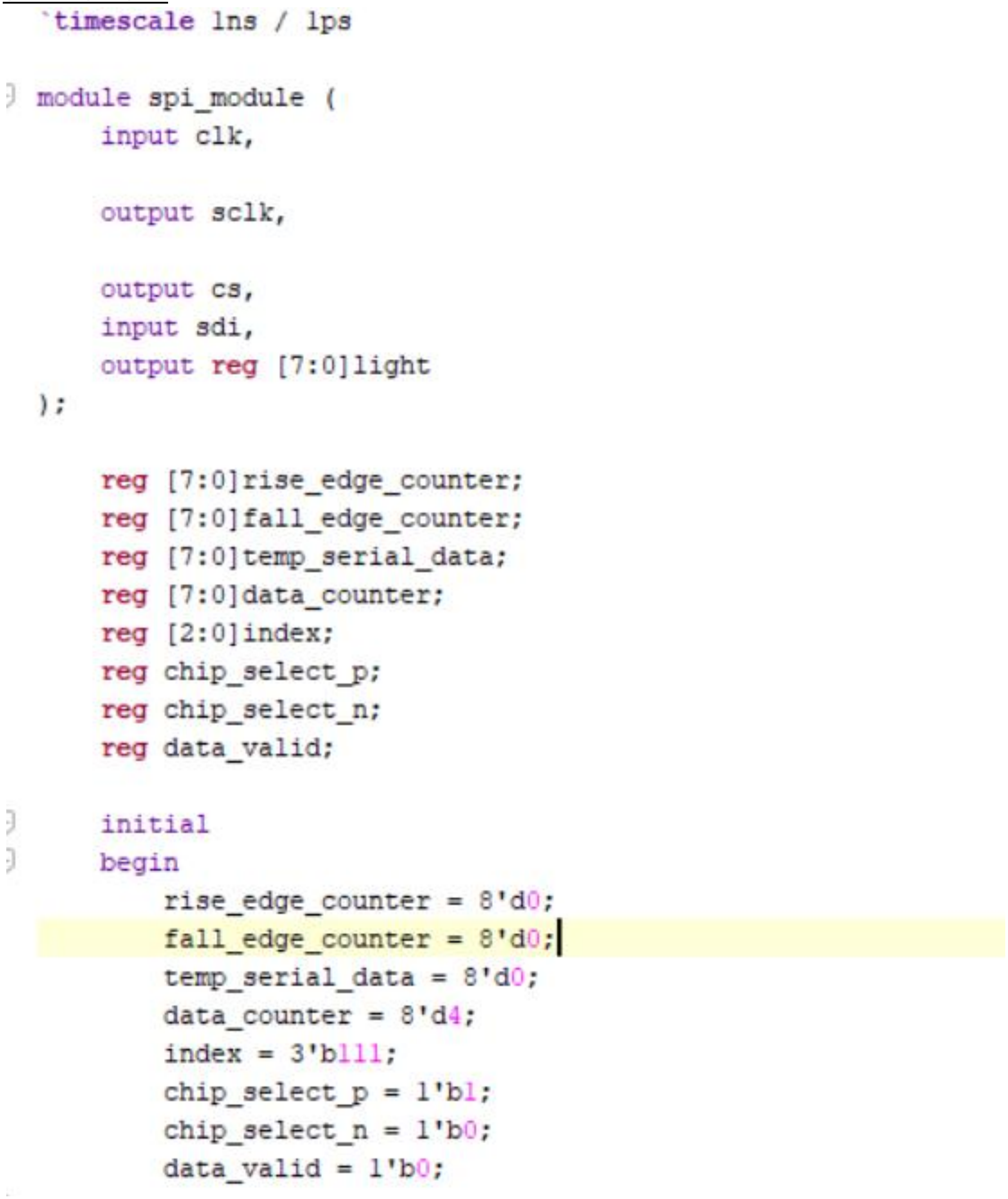
Clk DIVISION



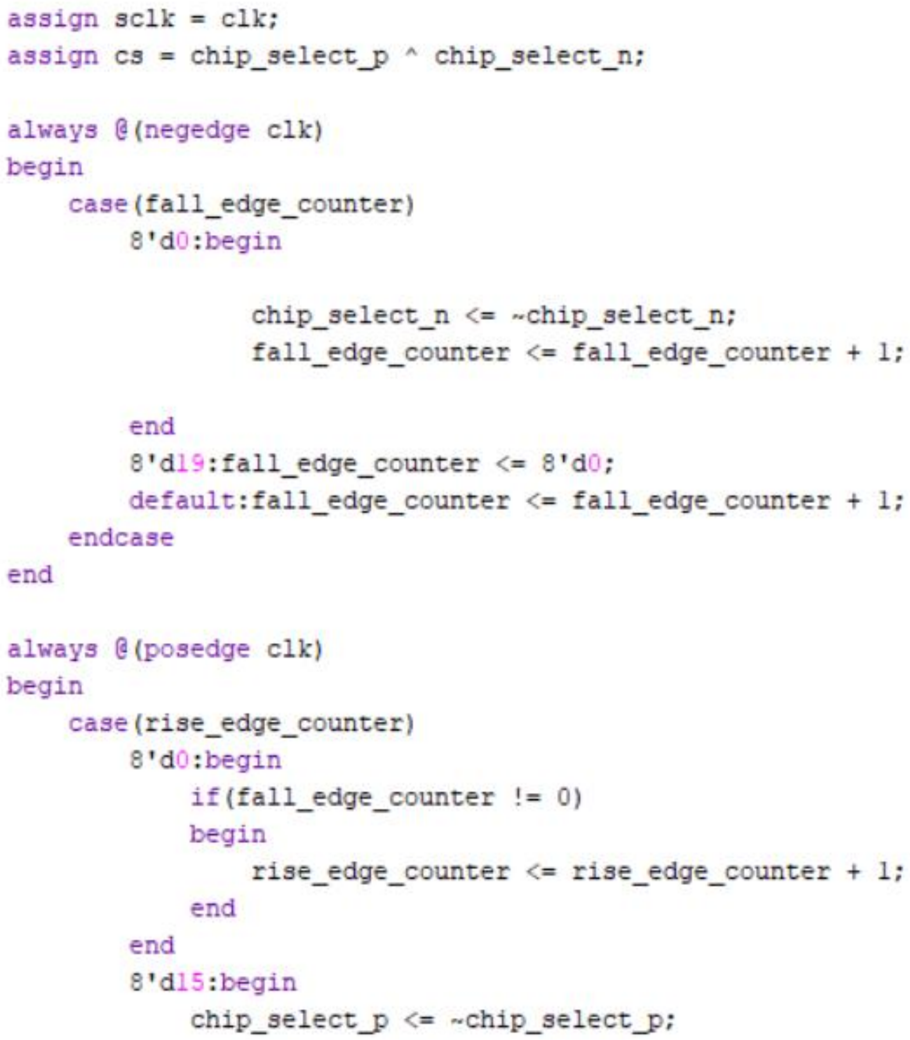
```
`timescale 1ns / 1ps

module clk_div (
    input clk_in,
    input reset,
    output clk_out
    );

    parameter N = 5;
    reg [N-1:0] clk_div;

    initial
    begin
        clk_div = 0;
    end

    always @(posedge clk_in, posedge reset)
    begin
        if(reset == 1)
            clk_div = 0;
        else
            clk_div = clk_div + 1;
    end

    assign clk_out = ~clk_div[N-1];

endmodule
```

We are using clock division to get into proper working range of pmod ALS which is 1-4MHZ here our clock frequency is 3.33MHZ which satisfies the condition

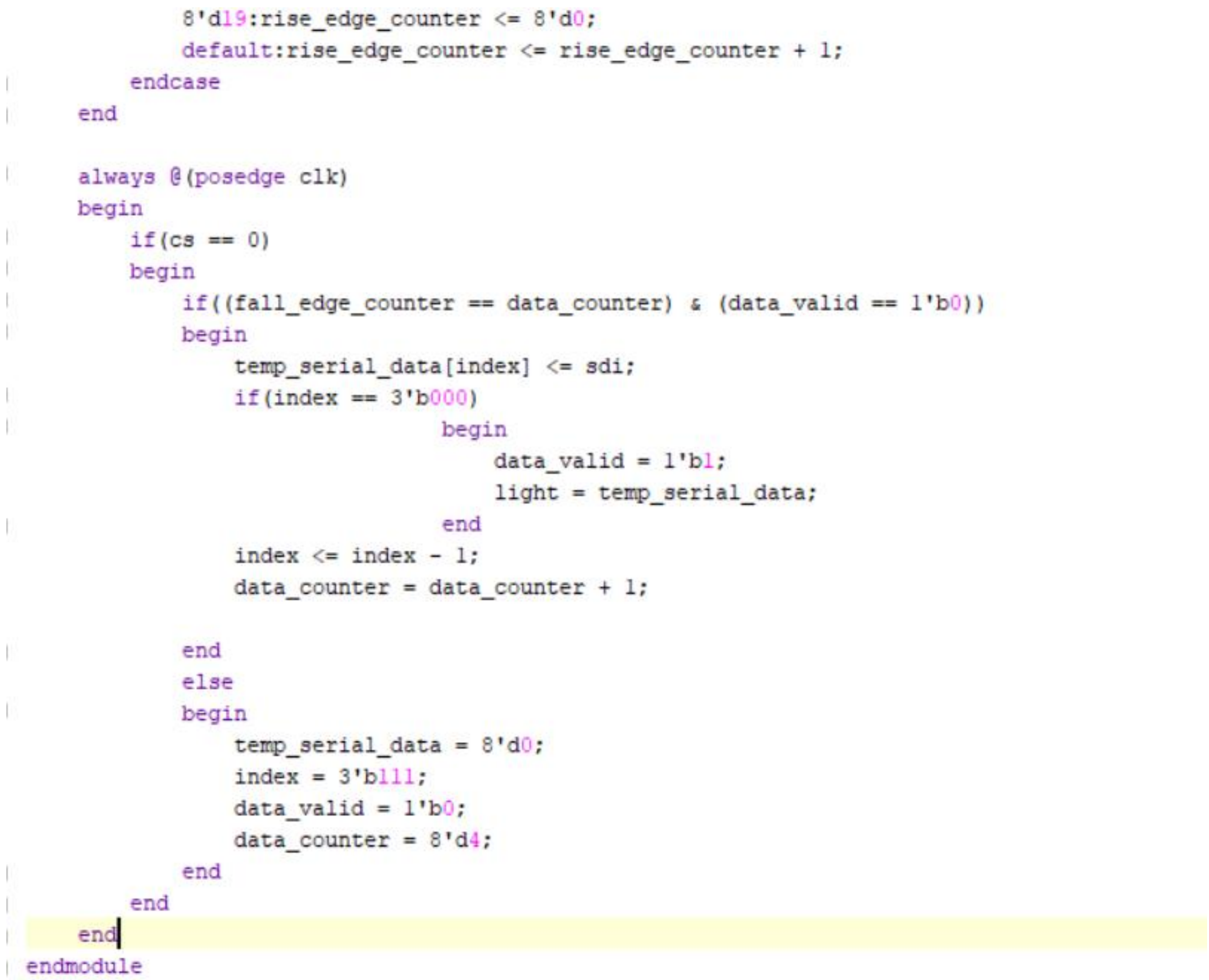
SPI MODULE



```
`timescale 1ns / 1ps

module spi_module (
    input clk,

    output sclk,

    output cs,
    input sdi,
    output reg [7:0]light
);

    reg [7:0]rise_edge_counter;
    reg [7:0]fall_edge_counter;
    reg [7:0]temp_serial_data;
    reg [7:0]data_counter;
    reg [2:0]index;
    reg chip_select_p;
    reg chip_select_n;
    reg data_valid;

    initial
    begin
        rise_edge_counter = 8'd0;
        fall_edge_counter = 8'd0;
        temp_serial_data = 8'd0;
        data_counter = 8'd4;
        index = 3'b111;
        chip_select_p = 1'b1;
        chip_select_n = 1'b0;
        data_valid = 1'b0;
```



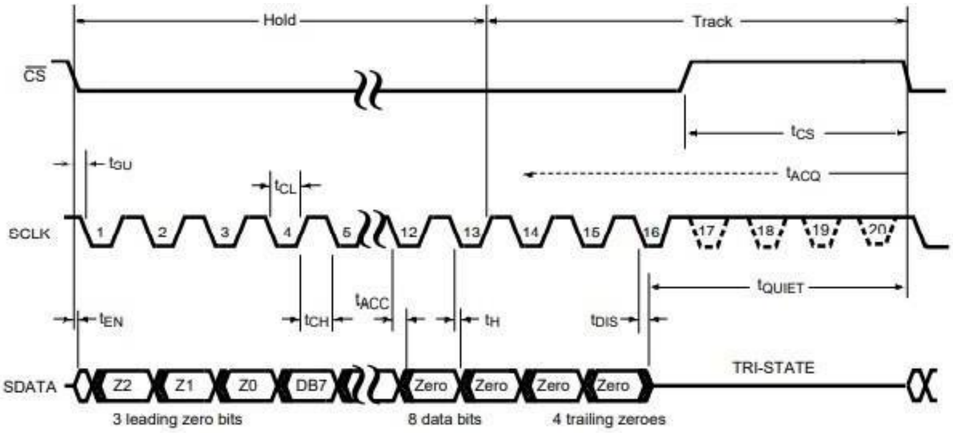
```
assign sclk = clk;
assign cs = chip_select_p ^ chip_select_n;

always @(negedge clk)
begin
    case(fall_edge_counter)
        8'd0:begin

                chip_select_n <= ~chip_select_n;
                fall_edge_counter <= fall_edge_counter + 1;

        end
        8'd19:fall_edge_counter <= 8'd0;
        default:fall_edge_counter <= fall_edge_counter + 1;
    endcase
end

always @(posedge clk)
begin
    case(rise_edge_counter)
        8'd0:begin
            if(fall_edge_counter != 0)
            begin
                rise_edge_counter <= rise_edge_counter + 1;
            end
        end
        8'd15:begin
            chip_select_p <= ~chip_select_p;
```



```
            8'd19:rise_edge_counter <= 8'd0;
            default:rise_edge_counter <= rise_edge_counter + 1;
        endcase
    end

    always @(posedge clk)
    begin
        if(cs == 0)
        begin
            if((fall_edge_counter == data_counter) & (data_valid == 1'b0))
            begin
                temp_serial_data[index] <= sdi;
                if(index == 3'b000)
                                begin
                                    data_valid = 1'b1;
                                    light = temp_serial_data;
                                end
                index <= index - 1;
                data_counter = data_counter + 1;

            end
            else
            begin
                temp_serial_data = 8'd0;
                index = 3'b111;
                data_valid = 1'b0;
                data_counter = 8'd4;
            end
        end
    end
endmodule
```

In this module we are getting intensity of light .

the sensor needs 16 clock cycles to send 15bits of data where first 3 bits and last 4 bits are leading and trailing zeros respectively. The exact timing of ADC081S021 is shown below.



THANK YOU
FOR THE OPPORTUNITY

FROM TEAM

ABHISHEK SAM
2019A3PS0363H

ARJUN REDDY SOMI REDDY
2019AAPS0301H

MANIKANTA SOMAYAJI
2019A3PS1339H
YARRAMSETTY SANJEEVA SAI PREETHAM
2019A3PS0485H