ELEX 7660

Now You See Me

Project Report

Ken Do, Andrew Obermeyer
4-14-2017

Ken Do
Andrew Obermeyer
April 14, 2017

Contents

Table of Figures

1 - Overview

1.1 - Project Motivation

Our street lighting requirements consume the largest amount of energy in our road infrastructure. The transition to more efficient light technology such as LED fixtures have improved the situation, but further reduction of energy consumption can only be realized by improving the lighting control systems.

The current control scheme turns on the lights at a set time or ambient light level. Once the set point is reached, the fixtures remain on at 100% output throughout the night. In locations with minimal to no traffic, these fixtures are consuming energy with no purpose. It is equivalent to leaving a light on in a vacant room overnight.

1.2 - Goals

Extending the lighting control systems ability to detect traffic will allow the system to provide sufficient light output when needed, and return to a low energy state once traffic has cleared. This is achieved by implementing the following system components:

- Use the FPGA's ADC to determine the output from a photo-resistor which will determine the amount of ambient light and trigger the device to enter night mode or day mode.
- Use the output from infrared sensors to detect motion on different sections of the modelled street. This could be accomplished using the ADC, external comparators, or directly using the outputs from the sensors as signals, if the voltage levels are consistent with logic levels.
- When at night and motion (a passing car) has been detected, turn on street lights where the car currently is as well as lights ahead of its path.
    - Automatically turn off these lights once the car has passed, by creating timers on the FPGA.

1.3 - System Block Diagram

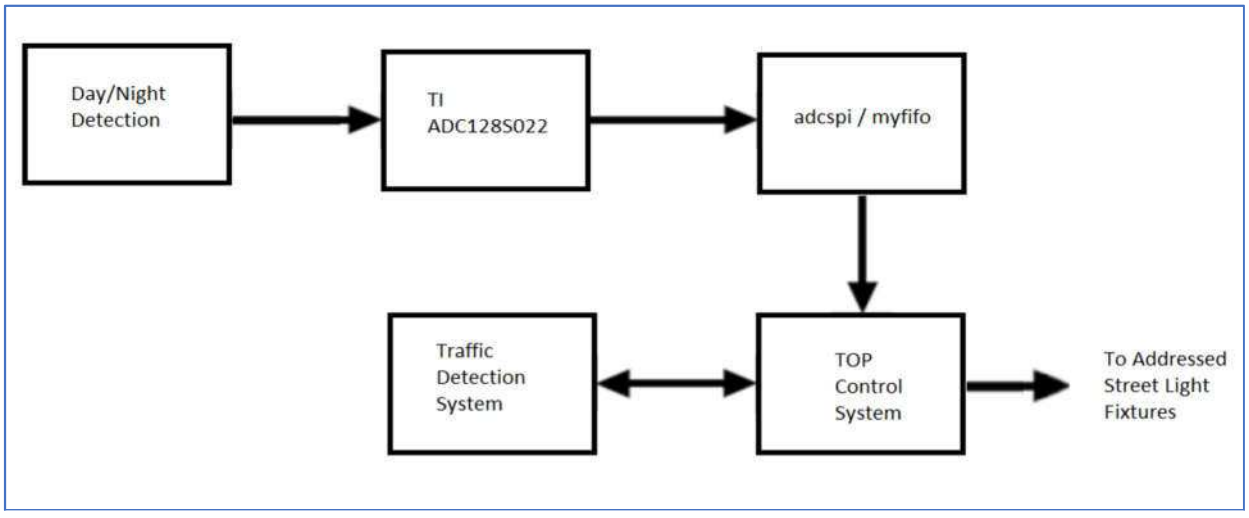

Figure 1: System Block Diagram.

1.4 - IP and Hardware Descriptions

Hardware / IP	Core Function	Creator(s)
CDS Photoconductive Photocell	• Photocell designed to sense light from 400 to 700 nm. • Dark resistance = 500 kΩ • Illuminated resistance = 33 kΩ	Luna Optoelectronics
Day/Night Detection System	• Simple resistor divider circuit containing a photocell. • Voltage output at the center of the resistor divider varies due to changes in light intensity detected by the photocell.	Andrew, Ken
TI ADC128S022	• Built-in ADC module used to read Day/Night Detection voltage output levels.	Texas Instruments
Adcspi	• Interfaces with the serial output of the built-in ADC module on the FPGA and stores the quantized data from channels 0 and 1.	Ed Casas
myfifo	• “First In First Out” interface to with adcspi.	Ken, Andrew
IR Emitter (LTR-4206E)	• Emits Infrared light with a wavelength of 940 nm	LITEON
IR Light to Voltage Converter (TLS267)	• Converts 940nm IR light intensity to output voltage	TAOS
Traffic Detection System	• Reads the voltage output from the IR detector associated to a street light location to detect traffic.	Andrew, Ken
Top Control System	• Sets street light output levels according to information provided by Day/Night and Traffic detection systems.	Ken, Andrew

2 - Outcome

2.1 - Results and Successes

The Day/Night detection system used a simple resistor divider circuit containing the photocell and was successful in detecting ambient lighting levels. The ADC quantization provided sufficient information to allow the programmer to set desired thresholds to distinguish between day and night scenarios.

The traffic detection system used the IR emitter/receiver pair to detect traffic. Each IR pair was addressed to match the light fixture location. The voltage output of the IR receiver provided a digital signal to the Top Control system. When sufficient IR light is detected, a high digital logic signal is sent to the Top Control system.

The Top Control system used the information provided by the previous two systems to control street light output. During day mode, all the street lights are deactivated. When night mode is detected, the street light output is determined by the traffic detection system. The traffic detection and the Timer algorithm within the Top Control system allowed for an active street lighting control system. When traffic is detected, the fixture in that location and the two sections ahead of that location are activated. This system successfully reduced the energy consumption of the street lights by only providing sufficient street lighting output when needed.

The Timer algorithm ensured the correct street lights were activated for a sufficient amount of time. The timer allowed for the street lights to remain active for a set period after traffic has cleared in that location. The traffic in that area will observe adequate lighting on the road ahead and behind them. The development of the timer algorithm within Quartus provided challenges due to Quartus not generating the expected hardware. Drawing out the state diagram helped produce the simplest code, which then generated the required hardware.

2.2 - Design Reviews

Our original plan was to use the ADC to determine the output level from the IR receivers, as initial testing determined that the ‘blocked’ and ‘unblocked’ voltages were approximately 0.6V and 1V respectively. We designed the system based on using one ADC channel to measure all 10 IR receivers, by only turning one transmitter/receiver pair on at a time and syncing this with the ADC conversion time. This was based on the incorrect assumption that the receiver outputs were high impedance. When this didn’t work, we tested the receiver outputs with a higher intensity IR from the transmitters, and determined that the ‘blocked’ and ‘unblocked’ output voltages were 0.3V and 3.2V respectively, when using the FPGA’s Vcc of 3.3V. This is sufficient for logic levels, and when combined with having all of the transmitters and sensors on 100% of the time rather than cycling through them, was used directly as the signal for the zone being tripped.

2.3 - Possibilities for Future Work

The model designed in this project serves as a basis for future expansion of features of a smart street light control system. Different scenarios could be implemented based on where the section of road is located. If the road is in a remote area, the auto-off system would be ideal as there would be limited vehicle traffic and no pedestrian traffic at night. In a more highly populated area, it might be more practical to have the lights dimmed rather than completely turned off when there is no traffic. This could simply be done using pulse-width modulation.

As stated previously, we attempted to use the ADC to measure the IR intensity at the receiver. Using the ADC would likely be the best way to implement this, as you could use any kind of IR sensor. This would require either using more ADC channels, or a means of isolating individual sensors.

The system as-built only supports one-way traffic, due to the way the timer module was designed to turn on the next two lights when a sensor is tripped. Upgrading to bi-directional movement would be possible with using an extra sensor at either end of the road segment to determine the direction of a moving vehicle.

3 - Description of Design Blocks

3.1 - Infrared Sensors

Infrared light sensors are commonly used for motion detection. While some, such as the yard light sensors found on many houses, are sensitive enough to pick up the infrared given off as body heat, others use a transmitter and receiver to create a stronger beam of infrared light whose intensity is determined by the receiver. This setup is commonly used as the safety sensor for automatic garage doors, to stop the door from closing if something or someone gets in the way, similar to Figure 3. We will use this technology as the motion sensors for our street light control system.

Figure 2: Infrared light can't be seen with the naked eye but can be as purple light with your phone's camera. This image was taken while cycling through the IR emitter/receiver pairs.

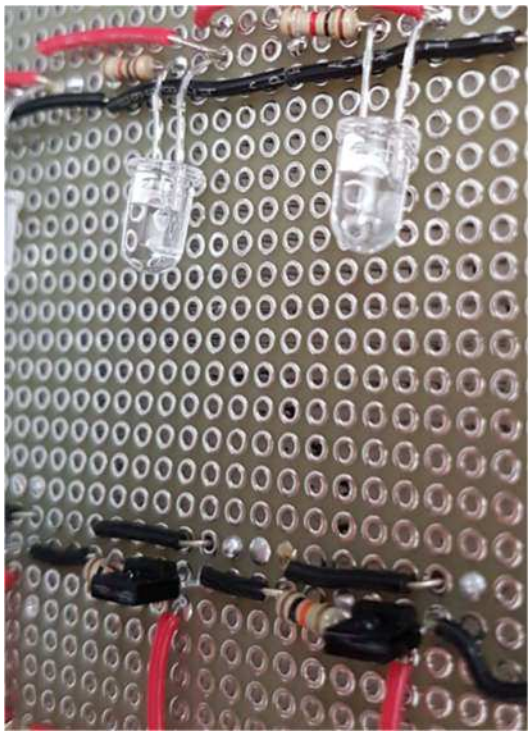

Figure 3: Infrared emitter LEDs (clear) directed at the infrared light detectors (black).

The IR sensors 'blocked' and 'unblocked' output voltages are approximately 0.3V and 3.2V respectively, when using the FPGA's Vcc of 3.3V. These voltage levels are sufficient for high and low logic levels, and are used as signals to directly drive the IR_LED_TxRx_Enable[i] signals for the light timer algorithm.

3.2 – Photo-resistor and ADC

The simple photo-resistor circuit shown in Figure 4 is widely used to detect ambient lighting levels due to its simplicity. The light intensity detected by the photocell changes its resistance. This allows the output voltage to vary with changes in light level intensity.

Using the built-in ADC on the FPGA board allowed for adjusting day/night detection threshold. The adjustability of this parameter enables this system to meet the requirements of different locations.

The ADC chip on the FPGA board was interfaced with the "adcspi.sv" module developed by Ed Casas. This module provides the ADC chip with signals for chip select, SCLK, and data in. It also receives the data out from the ADC. This module is configured to read ADC values from channels 0 and 1 on the ADC chip. Both values are stored as "data" register in the "adcspi.sv" module. Interfacing "myfifo.sv" module with "adcspi.sv" ensured that the first data conversion in was the first data conversion out.

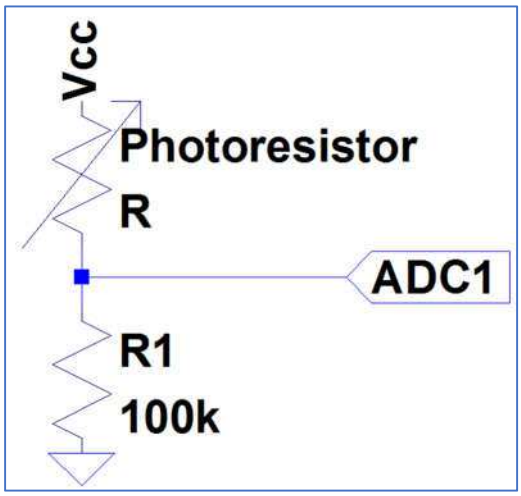

Figure 4: Photo-resistor circuit.

3.3 - Timer Algorithm

The auto-off timer control block is used to turn off individual street lights after they have they have been turned on by the IR_LED_TxRx_Enable[i] from the infrared receivers. The lights themselves are turned on by the scount[i] signal being greater than 0.

The timer algorithm can be visualized by the state transition diagram for the scount[i] signal below in Figure 5. Whenever a sensor is tripped, the value of ON_TIME (150,000,000 clock cycles, which is approximately 3 seconds), is loaded into scount[i] for the corresponding street light that is to be turned on. Because we wish to light the path ahead of the moving vehicle, the code also resets scount[i] for the next two street lights in series. The value of scount[i] is decremented every clock cycle, until it reaches 0, in which case it stays there. At any time, the value of scount[i] can be reset back to the value of ON_TIME, thus resetting the counter if the corresponding sensor or one of the previous two sensors are tripped.

The timer control block will only run if the operating mode is set to AUTO_OFF.

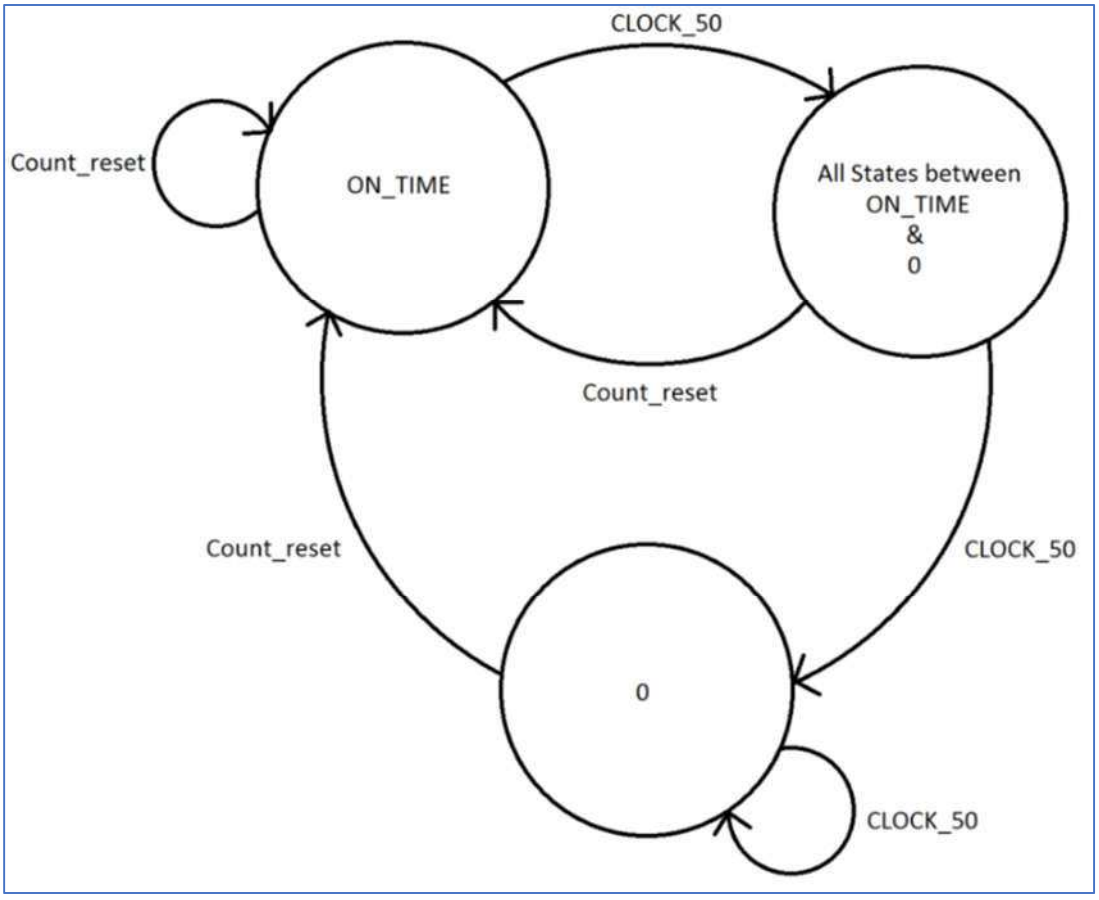

Figure 5: States of the scount[i] signal.

3.4 - Street Light Circuit Model

The street lights, modelled as a row of LEDs, are individually powered by the output signals Street_Light[i]. They use a 1kΩ resistor to control the current. The IR transmitter LEDs also use a 1kΩ resistor to limit current, but are all powered by the 3.3V Vcc of the FPGA. Both of these sets of LEDs are wired to common rails and connected to common on the FPGA. The IR receivers, Figure 6, are a 3-pin device, requiring Vcc and common from the FPGA. There is a 10kΩ resistor from output to common, and the output pin is also directly connected to the IR_LED_TxRx_Enable[i] input signal to the FPGA.

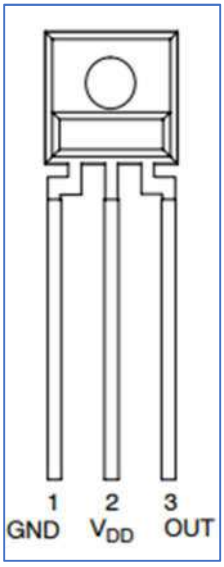

Figure 6: The TSL267 side-looker infrared receiver.

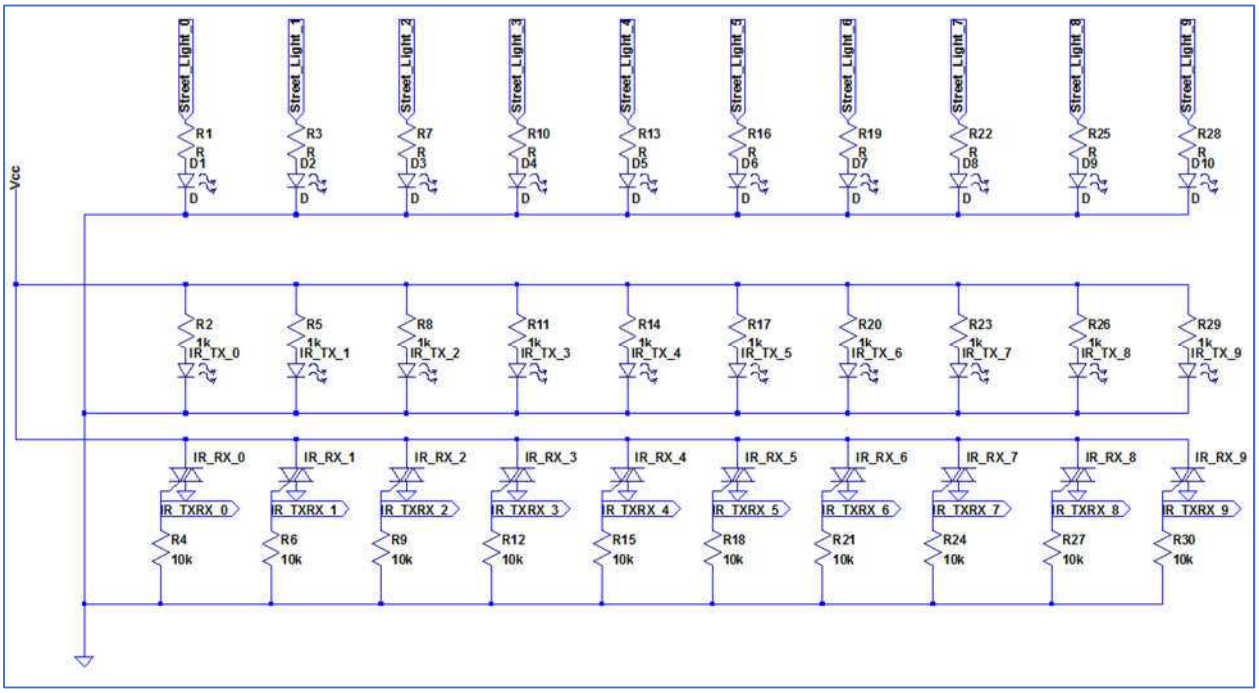

Figure 7: Circuit diagram of the model street light system.

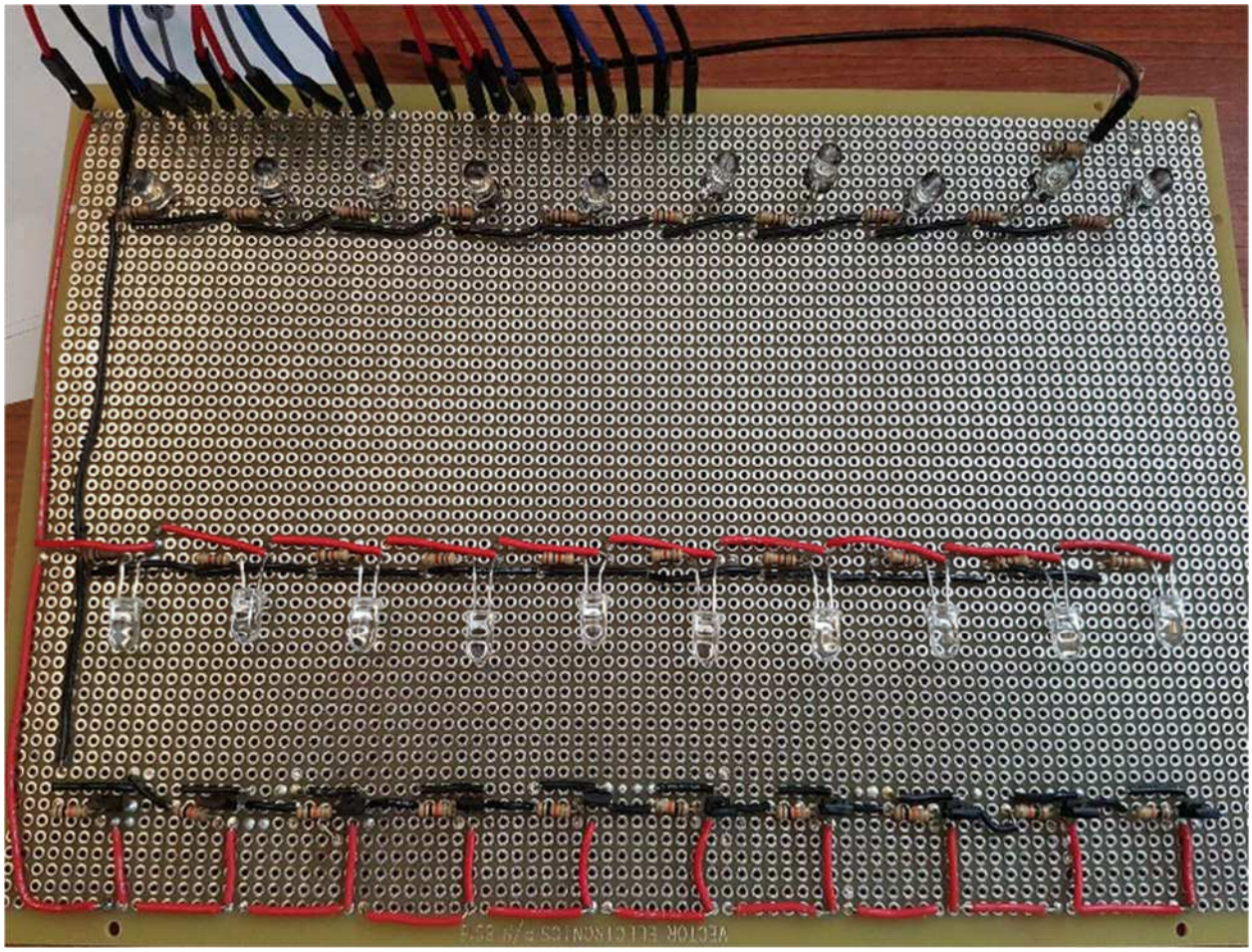

Figure 8: As-built model of the street light system containing rows of Street lights (top), IR emitter (middle), and IR receiver (bottom). V_{cc} are the red wires, common are the black wires. Photo-resistor circuit located on top right. FPGA connections are the row of wires on the top left.

4 - Systemverilog Code

4.1 - ADC.sv Module

```
// ADC.sv module - This module interfaces with the ADC of the FPGA board.
//              The ADC is used to detect day time status using a photocell.
// Modified by: Ken Do, Andrew Obermeyer
// Date:    April 9, 2017

`define MAXIRPAIR 9     // the largest address for IR pairs
`define CLOCK_DIV 10000 // Divisor for the ADC clock
`define THRESHOLD 300   // LED comparison threshold

module ADC
    (
     input  logic CLOCK_50,
     input  logic [1:0] KEY,        // reset_n

     output logic [3:0] IRPAIR,     // selects IR pair to be sampled.

     // GATE indicates the traffic status on the road. Bits 0 - MAXIRPAIR
     // represent the status of the IR sensors. Bit MAXIRPAIR + 1
     // represents the PHOTOCELL status.

     output logic [(`MAXIRPAIR+1):0] GATE,

     // ADC SPI interface
     output logic ADC_CS_N, // ssn
     output logic ADC_SADDR,    // mosi
     output logic ADC_SCLK, // sclk
     input  logic ADC_SDAT  // miso
);

   logic ready, valid, oready, ovalid;
   logic [31:0] data, odata;
   logic reset_n, clk;
   assign reset_n = KEY[0];

   // Divide module input clock by CLOCK_DIV.
   logic [31:0] count;
   always_ff @(posedge CLOCK_50) begin
      if (count <= 0) begin
         count <= `CLOCK_DIV;
         clk <= ~clk;
      end
      else count <= count - 1'b1;
   end


   adcspi a0
    (
     .sclk(ADC_SCLK),       // output to ADC clock
     .mosi(ADC_SADDR),      // output to ADC DIN (address)
     .ssn(ADC_CS_N),        // output to ADC chip select
     .miso(ADC_SDAT),       // input from ADC (DATA)
```

```
      .ready(ready),          // input
      .valid(valid),          // output
      .data(data),            // the ADC conversion data

      .clk(clk), .reset(~reset_n)
    );


// Photocell data control and output status.
    myfifo fifo0
     (
      .ready(ready),
      .valid(valid),
      .data(data),

      .oready(oready),
      .ovalid(ovalid),
      .odata(odata),
      .IRPAIR(IRPAIR),  // OUTPUT TO SYNCHRONIZE IR PAIR WITH ADC DATA
      .reset(reset_n),
      .clk(clk)
    );

    assign oready = '1;
    logic [11:0]  PHOTOCELL, TRAFFIC;
    assign PHOTOCELL = data[27:16];
    assign TRAFFIC = data[15:0];

    always_ff @(posedge clk)
     GATE[10] = (IRPAIR != 9) ? GATE[10] : (PHOTOCELL < 3000) ? 1'b1 : 1'b0;


endmodule

// -- start of adcspi.sv ---

// SPI master interface for TI ADC128S022
// for ELEX 7660 201710 Lab 5
// Ed.Casas 2017-2-16

// reads channels 0 and 1
// sclk is clk is divided by 16
// output is 16-bit samples from channels 0 and 1
// samples packed into 32 bits (ch 0 in MS byte)

// ADC128S0022 interface:
// 16 bit transfers

// mosi and cs* change on falling edge of sclk
// mosi bits 13:11 are (next) channel number

// miso sampled on rising edge of sclk
// miso data is on ls 12 bits of miso

// sample rate is sclk rate / 16
```

```
// sample rate must be 50 to 200 kHz
// sclk rate must be 800 kHz to 3.2 MHz
// e.g. 50 MHz / 32 = 1.5625 MHz sclk, ~98kHz sampling

// mosi timing relative to rising edge of sclk:
// setup is >10ns, hold  >10ns

// miso timing is relative to falling edge of sclk:
// access is <27ns, hold ~4ns

module adcspi
   (
    output logic sclk, mosi, ssn, // SPI master
    input logic miso,

    input logic ready,           // ready/valid data out
    output logic valid,
    output logic [31:0] data,

    input logic clk, reset
);

   parameter MISO = {5'b00001,27'b0} ;

   // clock/bit counter
   struct packed {
   logic wordcnt ;
   logic [3:0] bitcnt ;
   logic sclk ;
   logic [3:0] clkcnt ; } cnt, cnt_next ;

   logic [31:0] sr ;            // shift register

   logic rising, falling, done ;

   assign sclk = cnt.sclk ;

   // done all bits
   assign done = cnt ==? '{'1,'1,'1,'1} ;

   // clock/bit counter
   assign cnt_next = ( reset || done ) ? '0 : cnt+1'b1 ;
   always@(posedge clk)
     cnt <= cnt_next ;

   assign rising  =  cnt_next.sclk && ~cnt.sclk ;
   assign falling = ~cnt_next.sclk &&  cnt.sclk ;

   always@(posedge clk) begin

      if ( falling )            // shift mosi out
        mosi <= sr[31] ;

      if ( rising )             // shift miso in
        sr <= {sr[30:0],miso} ;
```

```
      if ( done ) begin
         data <= sr ;            // copy to parallel out
         sr <= MISO ;            // channel select serial out
         mosi <= MISO[31] ;
         valid <= '1 ;           // data ready
      end

      if ( ready && valid )      // data was read
        valid <= '0 ;

   end

   always@(posedge clk)          // run continously
     ssn <= reset ;

endmodule


// myfifo.sv - FIFO with ready/valid input and output
// for ELEX 7660 201710 lab 5
// Created by:      Ed Casas
// Modified by:     Ken Do, Andrew Obermeyer


module myfifo
       (
        output logic ready,        // ready/valid input
        input  logic valid,
        input  logic [31:0] data,

        input  logic oready,       // Avalon-ST output
        output logic ovalid,
        output logic [31:0] odata,
          output logic [3:0] IRPAIR,

        input logic reset, clk
);

   parameter W = 3 ;
   parameter N = 8 ;

   logic [31:0] DPRAM [7:0];                          // Dual-ported RAM
   logic [2:0] readp, readp_next,writep, writep_next;  // DPRAM in/out
pointers


   always_ff @(posedge clk) begin

    if (reset) begin
       readp  <= 3'b0;
           writep <= 3'b0;
       IRPAIR <= 4'b0;
        end
```

```
        if (ready) begin
       DPRAM[writep] <= data;

       if (IRPAIR >= 0 || IRPAIR < `MAXIRPAIR)
          IRPAIR <= IRPAIR + 1'b1;
       else
          IRPAIR <= 4'b0;
     end

        if (valid)
            writep <= writep_next;

        if (ovalid && oready)
            readp <= readp_next;
   end

   always_comb begin       // combinational logic for RAM status
   // input combinational logic
   ready = ((writep + 1'b1) != readp) ? 1'b1 : 1'b0;

   writep_next = (valid && ready) ? (writep + 1'b1) : writep;

   // output combinational logic

   ovalid = (readp != writep) ? 1'b1 : 1'b0;

   readp_next = (oready && ovalid) ? (readp + 1'b1) : readp;

   odata = DPRAM[readp];
    end

endmodule
```

4.2 - top.sv Module

```
// BCIT ELEX 7660 Final Project - Now You See Me
// Ken Do & Andrew Obermeyer


`define LED_COUNT 10            // # of LED sensors and lamps on street
model

`define DAY_MODE 0          // Day mode: all lights off
`define ALWAYS_ON 1         // Night mode: all lights always on full
intensity
`define AUTO_OFF 2          // Night mode: power saving, (auto) lights off
when no traffic
`define AUTO_DIM 3          // Night mode: power saving, (auto) lights dim
when no traffic

`define ON_TIME 32'd150000000       // 3 seconds (3 x 50MHz clock) = on time
for street lights when tripped


module top (input logic CLOCK_50,               // System clock

          input logic [1:0] KEY,                // Reset signal for ADC

          output logic ADC_CS_N,                // ADC chip select (active
low)
          output logic ADC_SCLK,                // ADC clock signal
          output logic ADC_SADDR,               // ADC control signal out
              input  logic ADC_SDAT,                    // ADC miso

          input logic [`LED_COUNT - 1:0] IR_LED_TxRx_Enable ,   // Input
from IR recievers
          output logic Street_Light [`LED_COUNT - 1:0],     // Output to
individual street lights

          output logic [7:0] LED  // on board LED's for testing
);


   logic [3:0] IRPAIR ;         // Input signal from ADC module to turn on
selected IR Transmitter/Reciever Pair
   logic [11:0] Lumen_Sensor;       // Lumen sensor result from 12-bit ADC
from lumen sensor
   logic [11:0] ADC_Result;     // Result of ADC conversion

   logic Street_Light_Mode_Enable [`LED_COUNT - 1:0];   // Street-Light
enable signal from mode selector block

   logic gate [`LED_COUNT - 1:0];   // IR beam broken (motion detected)
tracker

   logic [`LED_COUNT:0] GATE;       // Beam broken signal from ADC module
(bits 0 to 9) Night sensor (bit 10)
   // Please note that GATE[9:0] originally was intended to use the ADC to
determine whether the sensors were tripped
```

```
   // This was changed due to technical difficulties, and only GATE[10] was
used (for the photoresistor).

   logic [1:0] Operating_Mode;          // System operating mode:
day/night/power saving
   logic [31:0] scount [`LED_COUNT - 1:0];  // Counters for each light to
turn off
   logic count_reset [`LED_COUNT - 1:0];    // Count reset signal for timer
block


   ADC a1 (.*);              // Instantiate ADC module

   // Night/Day selector: change the night mode to one of 3 options based on
desired
   // operating preferences. See defines above for details.
   always_comb begin
      if (GATE[10]) begin       // GATE[10] is ADC result from ADC module
for the photoresistor
     Operating_Mode = `AUTO_OFF;    // Adjust this for operating mode
     LED[7] = 1;            // Turn on on-board LED #7 for verification
      end
      else begin
         Operating_Mode = `DAY_MODE;
     LED[7] = 0;
      end
   end

   // On-board LED's #0-6 for IR sensor verification
   assign LED[0] = ~IR_LED_TxRx_Enable[0];
   assign LED[1] = ~IR_LED_TxRx_Enable[1];
   assign LED[2] = ~IR_LED_TxRx_Enable[2];
   assign LED[3] = ~IR_LED_TxRx_Enable[3];
   assign LED[4] = ~IR_LED_TxRx_Enable[4];
   assign LED[5] = ~IR_LED_TxRx_Enable[5];
   assign LED[6] = ~IR_LED_TxRx_Enable[6];

   // Operating mode selector
   always_comb begin
      unique case(Operating_Mode)
      `DAY_MODE: begin      // Day mode - disable lights
     Street_Light_Mode_Enable[`LED_COUNT-1:0] = '{10{'0}};
      end
      `ALWAYS_ON: begin     // Night mode - enable all lights at full power
     Street_Light_Mode_Enable[`LED_COUNT-1:0] = '{10{1'b1}};
    end
      `AUTO_OFF: begin      // Power saving mode (auto off when no motion
detected)
     Street_Light_Mode_Enable[`LED_COUNT-1:0] = '{10{1'b1}};
      end
      `AUTO_DIM: begin      // Power saving mode (auto dim when no motion
detected) (not implemented)
     Street_Light_Mode_Enable[`LED_COUNT-1:0] = '{10{1'b1}};
      end
      endcase
```

```
    end


    // Auto-off timer control block
    always_ff @(posedge CLOCK_50) begin
        if(Operating_Mode == `AUTO_OFF) begin          // Only use if this mode
is enabled
      for(int i = 0; i < `LED_COUNT; i = i + 1) begin    // Generate hardware
for each light
          gate[i] <= ~IR_LED_TxRx_Enable[i];           // When a gate is
tripped, latch the signal
          if(gate[i]) begin                      // If a gate has been tripped
and latched:
              count_reset[i] <= 1;                   // Signal the count reset
for the light and the next 2 in series
              if(i < `LED_COUNT-1)                   // Stop generating hardware
if at the end of the road segment
            count_reset[i + 1] <= 1;
             if(i < `LED_COUNT-2)
            count_reset [i+2] <= 1;
              gate[i] <= 0;                      // Reset the gate-tripped latch
          end

          // Reset counter to starting position when gate is tripped, or count
down if count has been reset
               if(count_reset[i]) begin
              scount[i] <= `ON_TIME;
              count_reset[i] <= 0;          // Reset the count_reset signal
every time beam is triggered
          end
          else if(scount[i] > 0)            // Otherwise keep counting down if
count is not finished
              scount[i] <= scount[i] - 1;
          else
              scount[i] <= 0;               // Stay at 0 if count is finished
      end
       end
    end


    // Final enable signal to LED street lights. Must be enabled by both the
night/day sensor and the counters.
    // Signal from street light timers is the scount[i] signal. Lights will
only be on when this counter > 0.
    always_comb begin
        for(int i = 0; i < `LED_COUNT; i = i +  1)
      Street_Light[i] = (scount[i] > 0) & Street_Light_Mode_Enable[i];
    end


endmodule
```