**CSEN 605 :Digital System Design **

**Project Report**

**Bits Better Have My Money.**

|  |  |  |
| --- | --- | --- |
| **Member Name** | **Member ID** | **Tutorial #** |
| **Youssef Mohamed** | **34-3899** | **T-14** |
| **Mark Isaac** | **34-2024** | **T-17** |
| **Islam El-Gohary** | **34-5639** | **T-16** |
| **Bavly Emad** | **34-4463** | **T-16** |
| **Amr Hatem** | **34-4475** | **T-14** |
| **Hady Yasser** | **34-3051** | **T-10** |

**Smart Intersection**

**Submission Date :**

**16/12/2016**

1. **Overview:**

**Problem :**

One of the most irritating struggles in every Egyptian’s daily routine is the traffic congestion, and the time wasted every time we decide to hit the road, waiting in your car in a very congested road as the traffic lights are unfortunately red for your road, but green for that other intersecting road although it’s much less congested.

Also another common problem in Egypt is the lack of electricity and power shortage. The highways that are rarely used do use their road lights at night, even though it’s been more than an hour since a car have passed by.

We decided to offer an idea that will help in reducing and significantly affecting these problems.

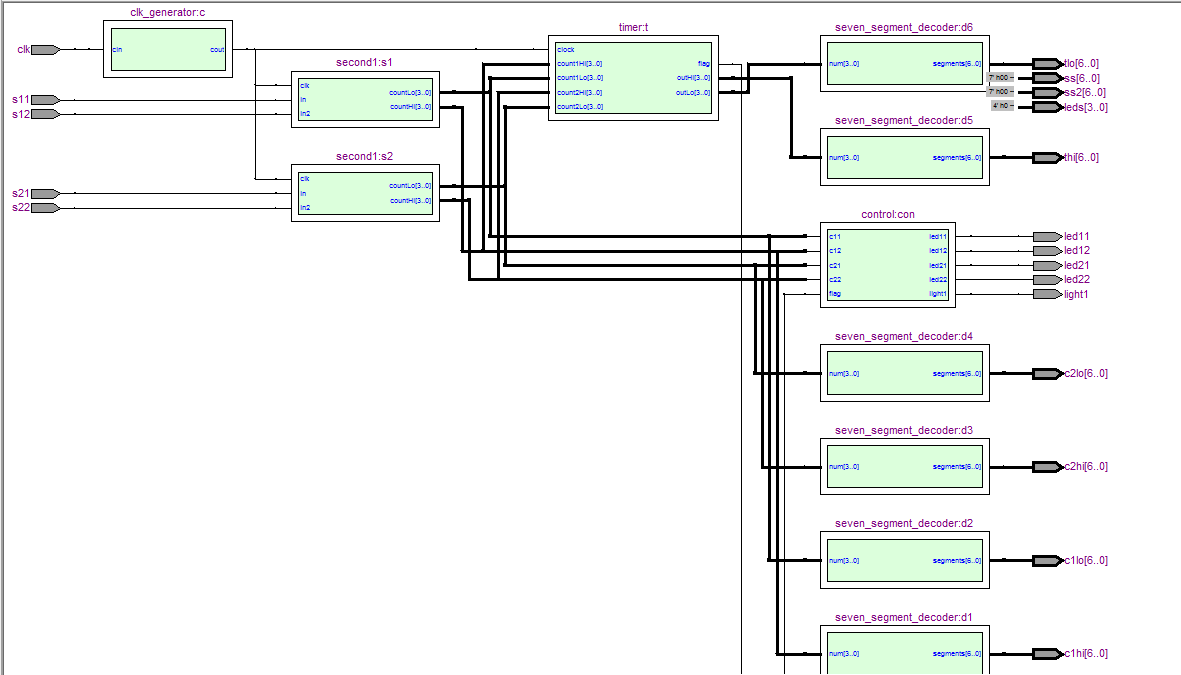
**Approach to Solve it :**

Instead of having the regular traffic lighting systems that only switch signals based on a constant timer that does not consider the severity of its road congestion, we’ve thought about coming up with an automated system that is smart enough to give priority to the more congested road in the intersection and accordingly giving it more time to breathe in order to balance the traffic flow.

The system will involve 2 sensors integrated in each road in the intersection, one in the source (Road entrance) and the other will be at the exit, we use these sensors to keep track of the number of cars present in each road at any given instance. Our system then adjusts the traffic Green/Red light duration having the degree of congestion in mind.

As for the electricity issue we simply make our system shut down the intersection lights when there are 0 cars detected in both road, and (Of course) re-enable it whenever there’s a driver passing by.

1. **Design Methodology**

**RTL Design:**

**Resources used:**

* ftp://ftp.altera.com/up/pub/Altera\_Material/Boards/DE2-115/DE2\_115\_User\_Manual.pdf
* [stackoverflow.com](http://stackoverflow.com)
* <http://www.alteraforum.com/forum/forum.php>

**Devices:**

Regarding the hardware we have:

* 4 I.R object detect sensors,
* 4 LEDs(2red and 2 green)and
* 8 100-ohm resistors.

every IRsensor is connected to the common ground and the common ground is connected to the FPGA Board and the power of sensors is connected to the 3.3Volt power source of the FPGA board and the outputs of the 4 sensors is connected to the input of the board and the output of the FPGA board is connected to the inputs of the 4 LEDs and each LED is connected to 2 series 100 ohm resistors

1. **Implementation:**

* **Sensor module :**

The Sensor module takes its inputs directly from Two sensors in each lane Those 2 sensors act as an increment and decrement Count-controls, in addition to the Clock input. It adds or subtracts 1 from the current car count. The outputs are 2 wires representing each digit in the car count with a maximum of 99 car per street.

* **Clock generator module :**

Takes as input the 50 MHz clock from the later board and delays it to 1 Hz.

* **Timer module :**

Takes as input the count of the cars in the 2 lanes and the delayed clock. It decides the timer start value when the traffic lights are switched according to the number of cars in the opening road (5 seconds per car).

The module keeps decrementing the timer every second until it reaches zero then it repeats the first step for the other road.

The module has 2 outputs:

* The Timer’s value which is displayed on the seven segment screen.
* A flag deciding which road is open right now.
* **Control module :**

The module takes as input the flag from the Timer and controls which traffic lights are on and which are off according to the value of the flag.

* **7-segment decoder module :**

Displays on the FPGA screen the timer for the traffic lights as well as the number of cars in each road.

* **Traffic control module :**

The traffic control module is like the big picture where all the connections between the inner modules are made.

1. **Code Listing:**

**Traffic module :**

Module traffic(clk,light1,s11,s12,s21,s22,led11,led12,led21,led22,c1lo,c1hi, c2lo,c2hi,tlo,thi,ss,ss2,leds);

input wire clk,s11,s12,s21,s22; // s11, s12, s21, s22 Are the 4 sensors from both roads.

output wire led11,led12,led21,led22,light1; // led11,led12,led21,led22 Are the Red/Green traffic lights for each road

//light1 is the lamp representing the street lights.

output reg [3:0]leds;

output wire[6:0] c1lo,c1hi,c2lo,c2hi,tlo,thi,ss,ss2;

wire clkhz;

wire flag;

wire statehi,statelo;

wire [3:0] c1hitemp, c2hitemp, c1lotemp,c2lotemp,thitemp,tlotemp;

clk\_generator c(clk,clkhz); // clock generator module

second1 s1(clkhz,s11,s12,c1lotemp,c1hitemp); // an instance of the sensor module for the first road

seven\_segment\_decoder d1 (c1hitemp,c1hi); // Seven sigment decoder module instance for the tens digit first road

seven\_segment\_decoder d2 (c1lotemp,c1lo); // Seven sigment decoder module instance for the units digit first road

second1 s2(clkhz,s21,s22, c2lotemp,c2hitemp); // an instance of the sensor module for the second road

seven\_segment\_decoder d3 (c2hitemp,c2hi); // Seven sigment decoder module instance for the tens digit second road

seven\_segment\_decoder d4 (c2lotemp,c2lo); // Seven sigment decoder module instance for the units digit second road

timer t(c1hitemp,c1lotemp,c2hitemp,c2lotemp,clkhz,thitemp, tlotemp,flag);

//Timer module instance

control con(c1lotemp,c1hitemp,c2lotemp,c2hitemp,flag,led11, led12,led21,led22,light1);

//Control module instance

seven\_segment\_decoder d5 (thitemp,thi);

//seven sigment decoder for the 10th digit displaying the time remaining till the green lights in the fpga

seven\_segment\_decoder d6 (tlotemp,tlo);

//seven sigment decoder for the units digit displaying the time remaining till the green lights in the fpga

//seven\_segment\_decoder d7 (statehi,ss2);

//seven\_segment\_decoder d8 (statelo,ss);

endmodule

**Clock generator module:**

module clk\_generator(cin,cout);

input cin; // the 50 MHz clock from the board

output reg cout= 0; //the delayed clock

reg [24:0] counter = 0; // Counter to decide when the output clock should be changed

always @(posedge cin)

begin

if (counter == 0) begin

counter <= 24999999; // counter updated for 25000000 times before cout toggles

cout <= ~cout; // after 50000000 times, The clock completes a full cycle

end else begin

counter <= counter -1; // decrement counter until it reaches zero

end

end

endmodule

**Timer module:**

module timer( count1Hi, count1Lo, count2Hi, count2Lo, clock,outHi , outLo,flag );

input[3:0] count1Hi; //tens digit of the count of the first road

input [3:0]count1Lo; // units digit of the count of the first road

input [3:0]count2Hi; //tens digit of the count of the second road

input[3:0] count2Lo;// units digit of the count of the second road

input clock; // 1Hz clock

output reg [3:0] outHi = 0; //tens digit of the timer

output reg [3:0] outLo = 0; // units digit of the timer

output reg flag=0; // flag determines which road is open

reg [9:0]out = 0; //timer in decimal

always@(posedge clock)

begin

if(outHi==0 && outLo==0) // if timer reaches zero

begin

if(flag) //if first road is to be opened

begin

out<=(count1Hi\*10+count1Lo)\*5; // calculate time according to count

end

else // if second road is to be opened

begin

out<=(count2Hi\*10+count2Lo)\*5;

end

if(out>90) // make sure that out does not exceed 90 seconds

out<=90;

outLo<=out%10; // put the units digit of the timer into outLo

outHi<=out/10;// put the tens digit of the timer into outHi

flag<=~flag; // Toggle the flag

end

else// decrement the timer

if(outLo>0) // if units is not 0 yet

outLo<=outLo-1; // decrement units

else if(outLo==0) // if units is 0

begin

outLo<=9; // make units = 9

outHi<=outHi-1; // decrement tens

end

end

endmodule

**Control Module:**

module control(flag,led11,led12,led21,led22);

input flag; // flag indicating which road is opened

output reg led11 = 1’b1; //initial value for the green light of the first road

output reg led12 = 1’b0; // initial value for the red light of the first road

output reg led21 = 1’b0; // initial value for the green light of the second road

output reg led22 = 1’b1; // initial value for the red light of the second road

always@(flag) // sensitive to the flag

begin

if(flag == 1)begin // if second road is opened

led11<=1’b0; // toggle the lights accordingly

led12<=1'b1;

led21<=1'b1;

led22<=1'b0;

end

else begin // if first road is opened

led11<=1’b1; // toggle the lights accordingly

led12<=1'b0;

led21<=1'b0;

led22<=1'b1;

end

end

endmodule

**Sesnors module :**

/\*Building on the modified clock module, We constructed a module whose only pusrpose in life is to listen to the insput signals from the sensors ,according to the positive edge of the clock, and either increment or decrement a certain counter variable which we later display on two 7-segments displayes. The counter is incremented or decremented based on which sensor is being cut. The detailed commented code below explains the nuts and bolts of this module.\*/

module second1 (input clk, in,in2,output reg[3:0] countLo,output reg[3:0] countHi);

// module header

reg state=0; // a reg variable that represents the state of the increment sensor

reg state2 =0; // a reg variable that represent the state of the decrement sensor

always @ (posedge clk)

// the start of the always block (which listens to the positive egde of the modifies (slowed) clock)

begin // similar to opening paranthesis in java and C/C++

if(!({in,in2} == 2'b00)) // check whether or not both sensors send a signal to the board at the same time (i.e: a car-

//- leaving the intersection while another one enters it at same time). if this happens, do nothing(because count shouldnt change), else proceed with code below

begin

case (state)

0: if(~in) begin state<=1;end // ( if the state of the increment sensor is 0 and we read a 0 from the sensor, we transition to state 1)

1: if(in) begin state<=0; // if the state of the increment sensor is 1 and we read a 1 we transition to state 0 and count based on the switch statement

case(countLo) // this switch statement block, checks for the current number (lower digit of the count)

// and based on its current value it decides its next value (in other words it simply counts)

4'b0000: begin countLo <= 4'b0001; end

4'b0001: begin countLo <= 4'b0010; end

4'b0010: begin countLo <= 4'b0011; end

4'b0011: begin countLo <= 4'b0100; end

4'b0100: begin countLo <= 4'b0101; end

4'b0101: begin countLo <= 4'b0110; end

4'b0110: begin countLo <= 4'b0111; end

4'b0111: begin countLo <= 4'b1000; end

4'b1000: begin countLo <= 4'b1001; end

4'b1001: //special case : if the lower digit reaches 9 , we increment the higher digit by 1 and set the lower digit to 0

begin countLo <= 4'b0000;

case(countHi) // repeats the same process done with the lower digit

4'b0000: begin countHi <= 4'b0001; end

4'b0001: begin countHi <= 4'b0010; end

4'b0010: begin countHi <= 4'b0011; end

4'b0011: begin countHi <= 4'b0100; end

4'b0100: begin countHi <= 4'b0101; end

4'b0101: begin countHi <= 4'b0110; end

4'b0110: begin countHi <= 4'b0111; end

4'b0111: begin countHi <= 4'b1000; end

4'b1000: begin countHi <= 4'b1001; end

4'b1001: begin countLo <= 4'b1001; end

default: begin countLo <= 4'b0000; end

endcase

end

default: begin countLo <= 4'b0000; end

endcase

end

endcase

case (state2) // the switch statement is identical to the one switching based on state on line 9 so only the difference will be commented

//the block listens to the decrement sensor instead of the increment sesnor ,but transitions in a similar manner

0: if(~in2) begin state2<=1;end

1: if(in2) begin state2<=0;

case(countLo) // this switch block checks the current number and decrements the lower digit based on the current value

4'b0000:

begin

countLo <= 4'b1001;

case(countHi) // here we handle a special case where the most significant digit hits 0

// when this happens we set the lower digit to be 9 and decrement the most signigicant one by 1

4'b0000: begin countHi <= 4'b0000; end

4'b0001: begin countHi <= 4'b0000; end

4'b0010: begin countHi <= 4'b0001; end

4'b0011: begin countHi <= 4'b0010; end

4'b0100: begin countHi <= 4'b0011; end

4'b0101: begin countHi <= 4'b0100; end

4'b0110: begin countHi <= 4'b0101; end

4'b0111: begin countHi <= 4'b0110; end

4'b1000: begin countHi <= 4'b0111; end

4'b1001: begin countHi <= 4'b1000; end

default: begin countHi <= 4'b0000; end

endcase

end

// end of the special case

4'b0001: begin countLo <= 4'b0000; end

4'b0010: begin countLo <= 4'b0001; end

4'b0011: begin countLo <= 4'b0010; end

4'b0100: begin countLo <= 4'b0011; end

4'b0101: begin countLo <= 4'b0100; end

4'b0110: begin countLo <= 4'b0101; end

4'b0111: begin countLo <= 4'b0110; end

4'b1000: begin countLo <= 4'b0111; end

4'b1001: begin countLo <= 4'b1000; end

default: begin countLo <= 4'b0000; end

endcase

end

endcase

end

end // similar to cloing paranthesis in java and C/C++

endmodule

/\*

further explanation to describe the 'state' and 'in' variables:

The sensor produces a digital output i.e: when the sesnor is cut by and object,

it sends a 0 (logical low) to the board. (active low sensor)

When that object moves away from the sesnor again, it produces 1.

When we read a 0 from the sesnors, since we are standing in state 0, based on the 1st two lines of the outer

most switch statement ,we move to state 1.This means that the next time we read a 1 from the sensor we will go

into the switch statement and either decrement or increment based on which sensor is being cut after setting the state back to 0

\*/

1. **Results and Limitations :**

After implementing the system circuit, we modeled the situation using a simple prototype is displayed in the video, it’s simulating the traffic flow in 2 roads crossing each other with the 2 sensors in each road, and 2 LED Traffic Green/Red lights for each road, in addition to 1 more LED on the FPGA modeling the Street lights.

We used an algorithm that multiplies the number of cars in each road by 5 and adjusts the Traffic lights accordingly (5 Seconds/Car).

Talking about the limitations, we can state that our design supports only :

- Intersections between 2 roads.

- Roads with a single lane, as each lane would require 2 sensors, an incrementer and a dectrementer.

- The Street Lights turn off when there’re no cars in BOTH roads.

**Who did What? :**

* Islam : The Control, Timer and Clock modules.
* Amr : The hardware implementation and connections to the board.
* Hady : The increment sensor in the sensors modules incrementing the count.
* Mark : The decrement sensor in the sensor modules decrementing the count.
* Bavly : Extending the idea of counters to work on two digit displays and their special cases.
* Youssef: Managing the inputs and outputs of all modules and bringing it together (Traffic Module).