Brittiney Sempasa

862155631

EE120A Section 21

Mini-Project: Traffic Light Controller

Ruth Navarrete

Overview:

In this project we will use RTL design methodology to design and implement a digital design for a 4-way traffic signal controller. Our goal is to create a stop light controller for a corner where two streets cross. A walker should be able to press a button and then, after the current state of the lights finishes, all lights should be in a red state for five seconds, allowing the walker to cross.

New concepts:

Register Transfer Level (RTL) design - is a method in which we can transfer data from one register to another orconstructing a digital design using Combinational and Sequential circuits in HDL like Verilog or VHDL which can model logical and hardware operation.

Analysis:

**BASIC DESIGN:**

High-Level State Machine

    Inputs: b(button), T(timer)

    Outputs: light1, light2, light\_w

    Local registers: b\_reg

|  |
| --- |
| b’T |

P1\_ON

P1\_INIT

|  |
| --- |
| b’T |

|  |
| --- |
| light1: 0 light2: 0  light\_w:0  reset: 1  cnt\_rst: 0  cnt\_int:0  b\_reg = 0 |

|  |
| --- |
| light1: 1 light2: 0  light\_w: 0  reset: 0  cnt\_rst: 3  cnt\_int: 0  b\_reg = b\_reg+b |

START

P2\_INIT

|  |
| --- |
| light1: 0 light2: 0  light\_w:0  reset: 1  cnt\_rst: 1  cnt\_int: 0  b\_reg = 0 |

|  |
| --- |
| light1: 0 light2: 0  light\_w:0  reset: 1  cnt\_rst: 0  cnt\_int:0  b\_reg = 0 |

|  |
| --- |
| b’T |

|  |
| --- |
| T |

|  |
| --- |
| bT |

|  |
| --- |
| T’ |

|  |
| --- |
| b’T’ |

WALK\_ON

WALK\_INIT

P2\_ON

|  |
| --- |
| bT |

|  |
| --- |
| light1: 0 light2: 0  light\_w: 0  reset: 1  cnt\_rst: 0  cnt\_int:0  b\_reg =0 |

|  |
| --- |
| light1: 0 light2: 1  light\_w:0  reset: 0  cnt\_rst: 3  cnt\_int: 0b\_reg = b\_reg+b |

|  |
| --- |
| light1: 0 light2: 0  light\_w: 1  reset: 0  cnt\_rst: 5  cnt\_int:0  b\_reg = b\_reg+b |

Finite State Machine

Inputs: b(button), clk(clock)

Outputs: light1, light2, light\_w, b\_cl, b\_ld

P1\_ON

(001)

P1\_INIT

(000)

(o

|  |
| --- |
| b’(T) |

|  |
| --- |
| light1: 0 light2: 0  light\_w: 0  b\_cl = 1  b\_ld= 0 |

|  |
| --- |
| light1: 1 light2: 0  light\_w: 0  b\_cl = 0  b\_ld= 1 |

START

P2\_INIT

(010)

|  |
| --- |
| light1: 0 light2: 0  light\_w: 0  b\_cl = 1  b\_ld= 0 |

|  |
| --- |
| light1: 0 light2: 0  light\_w: 0  b\_cl = 1  b\_ld= 0 |

|  |
| --- |
| b(T) |

|  |
| --- |
| T |

|  |
| --- |
| b’(T) |

|  |
| --- |
| T’ |

|  |
| --- |
| b’(T)’ |

WALK\_ON

(101)

WALK\_INIT

(100)

P2\_ON

(011)

|  |
| --- |
| light1: 0 light2: 0  light\_w: 0  b\_cl = 1  b\_ld= 0 |

|  |
| --- |
| light1: 0 light2: 1  light\_w: 0  b\_cl = 0  b\_ld= 1 |

|  |
| --- |
| light1: 0 light2: 0  light\_w: 1  b\_cl = 0  b\_ld= 1 |

|  |
| --- |
| b(T) |

**ADDITIONAL FEATURES:**

High-Level State Machine

    Inputs: b1(button),b2(button), T(timer)

    Outputs: light1(east/west lights), light2(north/south lights), light\_w, light\_t1(east/west/walk lights), light\_2(north/south/walk light)

    Local registers: b\_reg1, b\_reg2

|  |
| --- |
| (b1)’(b2)’T’ |

P1\_ON

P1\_INIT

|  |
| --- |
| (b1)’(b2)T |

|  |
| --- |
| (b1)’(b2)’T |

|  |
| --- |
| light1: 0 light2: 0  light\_w: 1  light\_t1:0  light\_t2: 1  reset: 0  cnt\_rst: 3  cnt\_int:0  b\_reg1 = b\_reg1+b1  b\_reg2 = b\_reg2+b2 |

|  |
| --- |
| light1: 0 light2: 0  light\_w: 0  light\_t1:0  light\_2: 0  reset: 1  cnt\_rst: 0  cnt\_int:0  b\_reg1 = 0  b\_reg2 = 0 |

START

P2\_INIT

|  |
| --- |
| light1: 0 light2: 0  light\_w: 0  light\_t1:0  light\_2: 0  reset: 1  cnt\_rst: 0  cnt\_int:0  b\_reg1 = 0  b\_reg2 = 0 |

|  |
| --- |
| light1: 0 light2: 0  light\_w: 0  light\_t1:0  light\_2: 0  reset: 1  cnt\_rst: 0  cnt\_int:0  b\_reg1 = 0  b\_reg2 = 0 |

|  |
| --- |
| (b1)’(b2)’T |

|  |
| --- |
| T |

|  |
| --- |
| (b1)’(b2)’T |

|  |
| --- |
| (b1)’(b2)’T |

|  |
| --- |
| T’ |

|  |
| --- |
| (b1)’(b2)’T’ |

|  |
| --- |
| T’ |

WALK\_ON1

WALK\_INIT1

P2\_ON

|  |
| --- |
| (b1)(b2)’T’ |

|  |
| --- |
| light1: 0 light2: 0  light\_w: 0  light\_t1:0  light\_t2: 0  reset: 1  cnt\_rst: 0  cnt\_int:0  b\_reg1 = 0  b\_reg2 = 0 |

|  |
| --- |
| light1: 0 light2: 0  light\_w: 1  light\_t1:0  light\_t2: 0  reset: 0  cnt\_rst: 5  cnt\_int:0  b\_reg1 = b\_reg1+b1  b\_reg2 = b\_reg2+b2 |

|  |
| --- |
| light1: 0 light2: 0  light\_w: 1  light\_t1:1  light\_t2: 0  reset: 0  cnt\_rst: 5  cnt\_int:0  b\_reg1 = b\_reg1+b1  b\_reg2 = b\_reg2+b2 |

|  |
| --- |
| (b1)’(b2)T |

|  |
| --- |
| light1: 0 light2: 0  light\_w: 1  light\_t1:0  light\_t2: 1  reset: 0  cnt\_rst: 5  cnt\_int:0  b\_reg1 = b\_reg1+b1  b\_reg2 = b\_reg2+b2 |

|  |
| --- |
| light1: 0 light2: 0  light\_w: 0  light\_t1:0  light\_2: 0  reset: 1  cnt\_rst: 0  cnt\_int:0  b\_reg1 = 0  b\_reg2 = 0 |

Oranges lines represent transitions to Walk(east/west/walk): (b1)(b2)’T

Blue lines represent transitions to Walk(north/east/walk): (b1)’(b2)T

Finite state machine

Inputs: b1(button),b2(button), clk(clock)

Outputs: light1, light2, light\_w, light\_t21, light\_t2, b\_cl1, b\_ld1, b\_cl2, b\_ld2

|  |
| --- |
| (b1)’(b2)’T’ |

P1\_ON

P1\_INIT

|  |
| --- |
| (b1)’(b2)T |

|  |
| --- |
| (b1)’(b2)’T |

|  |
| --- |
| light1: 1 light2: 0  light\_w: 0  light\_t1: 0  light\_t2: 0  b\_cl = 0  b\_ld= 1  b\_cl = 0  b\_ld= 1 |

|  |
| --- |
| light1: 0 light2: 0  light\_w: 0  light\_t1: 0  light\_t2: 0  b\_cl = 1  b\_ld= 0  b\_cl = 1  b\_ld= 0 |

START

P2\_INIT

|  |
| --- |
| light1: 0 light2: 0  light\_w: 0  light\_t1: 0  light\_t2: 0  b\_cl = 1  b\_ld= 0  b\_cl = 1  b\_ld= 0 |

|  |
| --- |
| light1: 0 light2: 0  light\_w: 0  light\_t1: 0  light\_t2: 0  b\_cl = 1  b\_ld= 0  b\_cl = 1  b\_ld= 0 |

|  |
| --- |
| (b1)’(b2)’T |

|  |
| --- |
| T |

|  |
| --- |
| (b1)’(b2)’T |

|  |
| --- |
| (b1)’(b2)’T |

|  |
| --- |
| T’ |

|  |
| --- |
| (b1)’(b2)’T’ |

|  |
| --- |
| T’ |

WALK\_ON1

WALK\_INIT1

P2\_ON

|  |
| --- |
| (b1)(b2)’T’ |

|  |
| --- |
| light1: 0 light2: 0  light\_w: 0  light\_t1: 0  light\_t2: 0  b\_cl = 1  b\_ld= 0  b\_cl = 1  b\_ld= 0 |

|  |
| --- |
| light1: 0 light2: 1  light\_w: 0  light\_t1: 0  light\_t2: 0  b\_cl = 0  b\_ld= 1  b\_cl = 0  b\_ld= 1 |

|  |
| --- |
| light1: 0 light2: 0  light\_w: 1  light\_t1: 0  light\_t2: 1  b\_cl = 0  b\_ld= 1  b\_cl = 0  b\_ld= 1 |

|  |
| --- |
| (b1)’(b2)T |

|  |
| --- |
| light1: 0 light2: 0  light\_w: 1  light\_t1: 0  light\_t2: 1  b\_cl = 0  b\_ld= 1  b\_cl = 0  b\_ld= 1 |

|  |
| --- |
| light1: 0 light2: 0  light\_w: 0  light\_t1: 0  light\_t2: 0  b\_cl = 1  b\_ld= 0  b\_cl = 1  b\_ld= 0 |

|  |
| --- |
| clk |

3

3

s

timer\_5

timer\_3

timer

B\_reg

B\_ld

B\_cl

Cnt\_init

Cnt\_rst

reset

clk

TIMER

B\_REG

STATE REGISTER

COMBINATIONAL

LOGIC

|  |
| --- |
| b |

CONTROLLER DATAPATH

light\_w

light2

light1

n

3

3

s

timer\_5

timer\_3

timer

Cnt\_init

Cnt\_rst

reset

clk

STATE REGISTER

COMBINATIONAL

LOGIC

**ADDITIONAL FEATURE:**

Buttons

light1, light2, light\_w, light\_t1, light\_t2

B\_ld2

B\_cl2

B\_reg2

B\_reg1

B\_cl1

B\_ld1

B\_REG1

TIMER

B\_REG2

clock

TOP LEVEL:

**module crosswalk**#(**parameter** NBITS = 32)

(

input wire b1,

input wire b2,

input wire clk,

output wire light1,

output wire light2,

output wire light\_w,

output wire light\_t1,

output wire light\_t2

)**;**

//internal wires

**wire** s0**;**

**wire** [NBITS-1:0] cnt\_ini**;**

**wire** reset**;**

**wire** b\_clr1**;**

**wire** b\_ld1**;**

**wire** b\_clr2**;**

**wire** b\_ld2**;**

**wire** timer**;**

**wire** b\_reg1**;**

**wire** b\_reg2**;**

crosswalk\_controller crosswalkcontroller(

.clk(clk),

.timer(timer),

.b\_reg1(b\_reg1),

.b\_reg2(b\_reg2),

.cnt\_ini(cnt\_ini),

.reset(reset),

.b\_clr1(b\_clr1),

.b\_ld1(b\_ld1),

.b\_clr2(b\_clr2),

.b\_ld2(b\_ld2),

.s0(s0),

.light1(light1),

.light2(light2),

.light\_w(light\_w),

.light\_t1(light\_t1),

.light\_t2(light\_t2)

);

crosswalk\_datapath crosswalkdatapath(

.s0(s0),

.cnt\_ini(cnt\_ini),

.reset(reset),

.clk(clk),

.b\_clr1(b\_clr1),

.b\_ld1(b\_ld1),

.b\_clr2(b\_clr2),

.b\_ld2(b\_ld2),

.b1(b1),

.b2(b2),

.b\_reg1(b\_reg1),

.b\_reg2(b\_reg2),

.timer(timer)

);

CONTROLLER:

**module crosswalk\_controller** #(**parameter** NBITS = 32)

(

input wire clk,

input wire b\_reg1,

input wire b\_reg2,

input wire timer,

output reg s0,

output reg [NBITS-1:0] cnt\_ini = 32'h0000,

output reg reset,

output reg b\_clr1,

output reg b\_ld1,

output reg b\_clr2,

output reg b\_ld2,

output reg light1,

output reg light2,

output reg light\_w,

output reg light\_t1,

output reg light\_t2

)**;**

**reg** [3:0] current\_state**;**

**reg** [3:0] next\_state**;**

// --------------------------------------

// Sequential logic

// --------------------------------------

**always** @(posedge clk) **begin**

current\_state <= next\_state ;

**end**

// --------------------------------------

// Comb. Logic - FSM

// --------------------------------------

localparam START = 4'b0000;

localparam P1\_INIT = 4'b0001;

localparam P1\_ON = 4'b0010;

localparam P2\_INIT = 4'b0011;

localparam P2\_ON = 4'b0100;

localparam WALK\_INIT\_1 = 4'b0101;

localparam WALK\_ON\_1 = 4'b0110;

localparam WALK\_INIT\_2 = 4'b0111;

localparam WALK\_ON\_2 = 4'b1000;

**always** @( current\_state ) **begin**

**case** **(**current\_state**)**

START **:** begin

light1 = 1'b0**;**

light2 = 1'b0**;**

light\_w = 1'b0**;**

light\_t1 = 1'b0**;**

light\_t2 = 1'b0**;**

s0 = 1'b0**;**

reset = 1'b1**;**

b\_clr1 = 1'b1**;**

b\_ld1 = 1'b0**;**

b\_clr2 = 1'b1**;**

b\_ld2 = 1'b0**;**

next\_state = P1\_INIT**;**

end

P1\_INIT **:** begin

light1 = 1'b0**;**

light2 = 1'b0**;**

light\_w = 1'b0**;**

light\_t1 = 1'b0**;**

light\_t2 = 1'b0**;**

s0 = 1'b0**;**

reset = 1'b1**;**

b\_clr1 = 1'b1**;**

b\_ld1 = 1'b0**;**

b\_clr2 = 1'b1**;**

b\_ld2 = 1'b0**;**

next\_state = P1\_ON**;**

end

P1\_ON **:** begin

light1 = 1'b1**;**

light2 = 1'b0**;**

light\_w = 1'b0**;**

light\_t1 = 1'b0**;**

light\_t2 = 1'b0**;**

s0 = 1'b0**;**

reset = 1'b0**;**

b\_clr1 = 1'b0**;**

b\_ld1 = 1'b1**;**

b\_clr2 = 1'b0**;**

b\_ld2 = 1'b1**;**

if**(**timer&&~b\_reg1&&~b\_reg2**)** begin

next\_state = P2\_INIT**;**

end

else if **(**timer&&b\_reg1&&~b\_reg2**)** begin

next\_state = WALK\_INIT\_1**;**

end

else if **(**timer&&~b\_reg1&&b\_reg2**)** begin

next\_state = WALK\_INIT\_2**;**

end

else begin

next\_state = P1\_ON**;**

end

end

P2\_INIT **:** begin

light1 = 1'b0**;**

light2 = 1'b0**;**

light\_w = 1'b0**;**

light\_t1 = 1'b0**;**

light\_t2 = 1'b0**;**

s0 = 1'b0**;**

reset = 1'b1**;**

b\_clr1 = 1'b1**;**

b\_ld1 = 1'b0**;**

b\_clr2 = 1'b1**;**

b\_ld2 = 1'b0**;**

next\_state = P2\_ON**;**

end

P2\_ON **:** begin

light1 = 1'b0**;**

light2 = 1'b1**;**

light\_w = 1'b0**;**

light\_t1 = 1'b0**;**

light\_t2 = 1'b0**;**

s0 = 1'b0**;**

reset = 1'b0**;**

b\_clr1 = 1'b0**;**

b\_ld1 = 1'b1**;**

b\_clr2 = 1'b0**;**

b\_ld2 = 1'b1**;**

if**(**timer&&~b\_reg1&&~b\_reg2**)** begin

next\_state = P1\_INIT**;**

end

else if **(**timer&&b\_reg1&&~b\_reg2**)** begin

next\_state = WALK\_INIT\_1**;**

end

else if **(**timer&&~b\_reg1&&b\_reg2**)** begin

next\_state = WALK\_INIT\_2**;**

end

else begin

next\_state = P2\_ON**;**

end

end

WALK\_INIT\_1 **:** begin

light1 = 1'b0**;**

light2 = 1'b0**;**

light\_w = 1'b0**;**

light\_t1 = 1'b0**;**

light\_t2 = 1'b0**;**

s0 = 1'b1**;**

reset = 1'b1**;**

b\_clr1 = 1'b1**;**

b\_ld1 = 1'b0**;**

b\_clr2 = 1'b1**;**

b\_ld2 = 1'b0**;**

next\_state = WALK\_ON\_1**;**

end

WALK\_ON\_1 **:** begin

light1 = 1'b0**;**

light2 = 1'b0**;**

light\_w = 1'b1**;**

light\_t1 = 1'b1**;**

light\_t2 = 1'b0**;**

s0 = 1'b1**;**

reset = 1'b0**;**

b\_clr1 = 1'b0**;**

b\_ld1 = 1'b1**;**

b\_clr2 = 1'b0**;**

b\_ld2 = 1'b1**;**

if **(**timer**)** begin

next\_state = P1\_INIT**;**

end

else begin

next\_state = WALK\_ON\_1**;**

end

end

WALK\_INIT\_2 **:** begin

light1 = 1'b0**;**

light2 = 1'b0**;**

light\_w = 1'b0**;**

light\_t1 = 1'b0**;**

light\_t2 = 1'b0**;**

s0 = 1'b1**;**

reset = 1'b1**;**

b\_clr1 = 1'b1**;**

b\_ld1 = 1'b0**;**

b\_clr2 = 1'b1**;**

b\_ld2 = 1'b0**;**

next\_state = WALK\_ON\_2**;**

end

WALK\_ON\_2 **:** begin

light1 = 1'b0**;**

light2 = 1'b0**;**

light\_w = 1'b1**;**

light\_t1 = 1'b0**;**

light\_t2 = 1'b1**;**

s0 = 1'b1**;**

reset = 1'b0**;**

b\_clr1 = 1'b0**;**

b\_ld1 = 1'b1**;**

b\_clr2 = 1'b0**;**

b\_ld2 = 1'b1**;**

if **(**timer**)** begin

next\_state = P1\_INIT**;**

end

else begin

next\_state = WALK\_ON\_2**;**

end

end

**default:** begin

light1 = 1'b0**;**

light2 = 1'b0**;**

light\_w = 1'b0**;**

light\_t1 = 1'b0**;**

light\_t2 = 1'b0**;**

s0 = 1'b0**;**

reset = 1'b1**;**

b\_clr1 = 1'b1**;**

b\_ld1 = 1'b0**;**

b\_clr2 = 1'b1**;**

b\_ld2 = 1'b0**;**

next\_state = P1\_INIT**;**

end

**endcase**

**end**

endmodule

DATAPATH:

**module crosswalk\_datapath** #(**parameter** NBITS = 32)

(

input wire s0,

input wire [NBITS-1:0] cnt\_ini,

input wire reset,

input wire clk,

input wire b\_clr1,

input wire b\_ld1,

input wire b1,

input wire b\_clr2,

input wire b\_ld2,

input wire b2,

output wire timer,

output wire b\_reg1,

output wire b\_reg2

)**;**

**reg** [31:0] t3 = 32'h8f0d180**;**

**reg** [31:0] t5 = 32'hee6b280**;**

**wire** [NBITS-1:0] cnt\_rst**;**

// --------------------------------------

// B\_reg1

// ---------------------------------------

b\_reg1 breg1(

.b\_clr1(b\_clr1),

.b\_ld1(b\_ld1),

.b1(b1),

.clk(clk),

.b\_reg1(b\_reg1)

);

// --------------------------------------

// B\_reg2

// ---------------------------------------

b\_reg2 breg2(

.b\_clr2(b\_clr2),

.b\_ld2(b\_ld2),

.b2(b2),

.clk(clk),

.b\_reg2(b\_reg2)

);

// --------------------------------------

// MUX

// --------------------------------------

crosswalk\_mux crosswalkmux(

.s0(s0),

.t3(t3),

.t5(t5),

.cnt\_rst(cnt\_rst)

);

// --------------------------------------

// Timer instantiation

// --------------------------------------

timer\_bh timerbh(

.reset(reset),

.clk(clk),

.cnt\_ini(cnt\_ini),

.cnt\_rst(cnt\_rst),

.timer(timer)

);

endmodule

B\_REGISTER\_1:

**module b\_reg1** #(**parameter** NBITS = 32)

(

input wire b\_clr1,

input wire b\_ld1,

input wire b1,

input wire clk,

output reg b\_reg1

)**;**

**always**@(posedge clk) **begin**

**if**(b\_clr1)**begin**

b\_reg1 = 0;

**end**

**else** **if**(b\_ld1) **begin**

b\_reg1 = b\_reg1 + b1;

**end**

**else** **begin**

b\_reg1 = b\_reg1;

**end**

**end**

endmodule

B\_REGISTER\_2:

**module b\_reg2** #(**parameter** NBITS = 32)

(

input wire b\_clr2,

input wire b\_ld2,

input wire b2,

input wire clk,

output reg b\_reg2

)**;**

**always**@(posedge clk) **begin**

**if**(b\_clr2)**begin**

b\_reg2 = 0;

**end**

**else** **if**(b\_ld2) **begin**

b\_reg2 = b\_reg2 + b2;

**end**

**else** **begin**

b\_reg2 = b\_reg2;

**end**

**end**

endmodule

MUX:

**module crosswalk\_mux** #(**parameter** NBITS = 32)

(

input wire s0,

input wire [NBITS-1:0] t3,

input wire [NBITS-1:0] t5,

output reg [NBITS-1:0] cnt\_rst

)**;**

**always**@(s0) **begin**

**case(**s0**)**

1'b0 **:** begin

cnt\_rst = t3**;**

end

1'b1 **:** begin

cnt\_rst = t5**;**

end

**default** **:** begin

cnt\_rst = t3**;**

end

**endcase**

**end**

endmodule

TIMER:

**module timer\_bh** (

**input** wire clk,

input wire reset,

input wire [31:0] cnt\_ini,

input wire [31:0] cnt\_rst,

output reg timer

)**;**

**reg** [31:0] count = 32'h0000**;**

**always**@(posedge clk) **begin**

**if**(reset)**begin**

count = cnt\_ini;

timer = 1'b0;

**end**

**else** **if** (count == cnt\_rst) **begin**

count = cnt\_ini;

timer = 1'b1;

**end**

**else** **begin**

count = count + 1;

timer = 1'b0;

**end**

**end**

endmodule

TESTBENCH:

**module crosswalk1\_tb**;

// Inputs

**reg** b1**;**

**reg** b2**;**

**reg** clk**;**

// Outputs

**wire** light1**;**

**wire** light2**;**

**wire** light\_w**;**

**wire** light\_t1**;**

**wire** light\_t2**;**

// Instantiate the Unit Under Test (UUT)

crosswalk uut (

.b1(b1),

.b2(b2),

.clk(clk),

.light1(light1),

.light2(light2),

.light\_w(light\_w),

.light\_t1(light\_t1),

.light\_t2(light\_t2)

);

initial **begin**

clk = 0;

forever **begin**

#50 clk = ~clk;

**end**

**end**

initial **begin**

b1 = 0;

forever **begin**

#50 b2 = ~b2;

**end**

**end**

initial **begin**

b2 = 0;

forever **begin**

#50 b2 = ~b2;

**end**

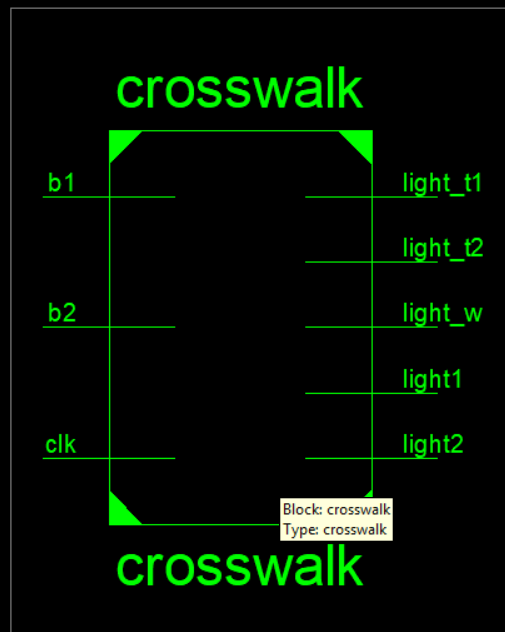
**end**

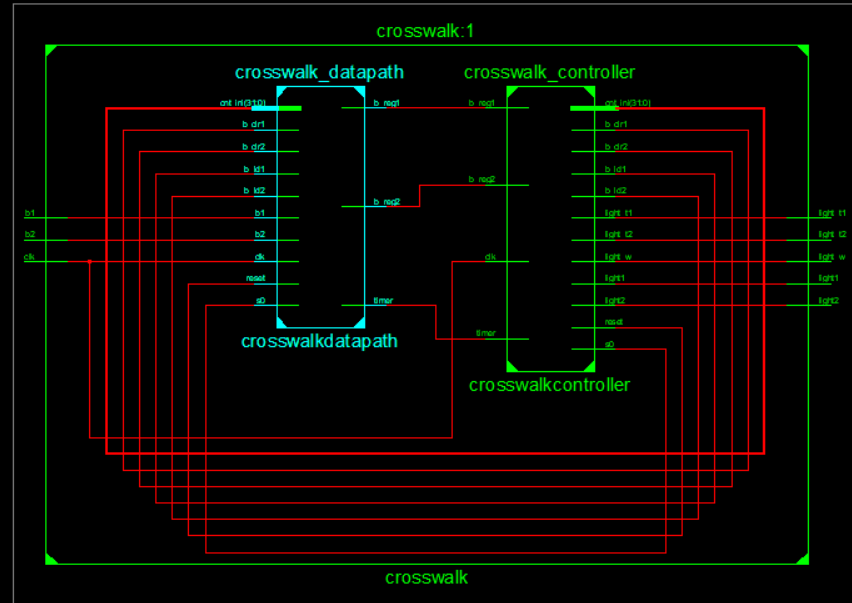
**endmodule**

Records:

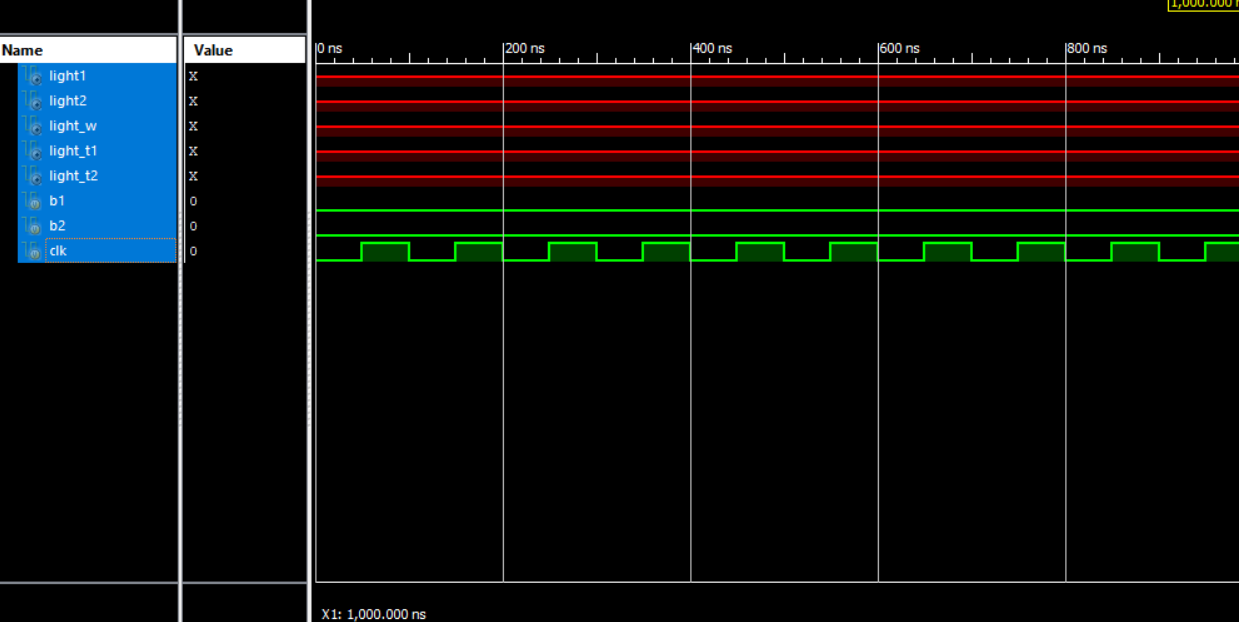
ADDITIONAL FEATURES:

TOP LEVEL RTL:





TESTBENCH WAVEFORM:



Youtube demonstration:

<https://www.youtube.com/watch?v=JbVC-KpRia4>

Discussion:

By far the most difficult part of the mini-project was understanding how to implement the state machine onto Xilinx. We approached the project as a coding assignment rather than wiring components to ports. It grew easier if we just knew what each thing was doing. We had originally used the same timer from lab 6 but it grew complicated and was suggested to create a new timer behaviorally. We ended up doing each component behaviorally and the top module was done structurally.

We were unable to get the testbench to run correctly as there are two reasons that it may not run. Verilog must have the output initialized to a value or the clocks aren’t syncing up together leading to a timing issue. We double checked to see if our output lights were initialized and in our controller module all output lights are initialized to a binary value of zero. We had also tested other ways of initializing the value to 0 from using 1’b0 and the value of 0 and we experienced the same result. Once we saw that the initialization wasn’t the cause, we checked the timing. We tried using the time scale given 1ns/1ps and adjusting it to 1ns/ 500ns and 1ns/5000ns. Our lights still didn’t output anything. There was no warning given that could have affected our testbench such as flip flop errors. We still believe that maybe our timing scale is still off as the code was cross checked with our TA, Sherrif, current classmates, and people who had previously taken the class.

Conclusion:

Overall, we were able to create the high-level state machine, convert it to a finite state machine, and implement the design on Xilinx and the fpga board successfully. Our drawn RTL design as reflected the RTL design given from Xilinx. We unfortunately couldn’t get the testbench of the top module to work and are still unsure why it isn’t working. The system was a bit over complicated as we had added extra lights to simulate the crosswalk and street light being active when we could have used the 3 lights we originally had and made one of the lights 1 rather than 0 and alternate based off which button was pressed.

Ways to improve the project would have been adding turn signals or having it remember which state it was and continuing the cycle like a normal traffic light. For example, if I had my North/South crosswalk activated the North/South crosswalk and street lights would be on and the East/West lights would be off and once it finished it would go to the East/West street lights and visa versa. In our case we had it start back at the first state regardless of which button was pressed.