CS/EE 120A
(Section: 021)

Lab #6:
Laser Surgery System

Dan Murphy; SID: 862xxxxxx
dmurp006@ucr.edu

Partner:
Jesse Garcia

Subject Overview

This lab jumped straight into the fire, skipping the first two parts (A & B) and diving right into Part C.

Part C:

Within this section, our goal was to create a system which outputs an LED onto our BASYS FPGA board for 5 seconds when a button is pressed. In order to do this, we had to create transitions for our FSM (Finite State Machine): OFF, START & ON. This was done through cases within Verilog.

OFF: Infinite looping recurrence until a button is pressed, which it then triggers its respective cases START -> ON.

START: Transitional state with respect to ON.

ON: Triggers light to stay on. Light will remain illuminated until the timer is equal to 1.

New Concepts

1. Describe how this lab built upon previous ones:

This lab built upon previous labs as we have to understand how to design and synthesize counters into Verilog and further grasp the concept of utilizing clocks.

2. Describe the most difficult part of this lab for you:

By far, the most difficult part was realizing that this was a simple lab. Once I broke down the logic of the lab assignment on paper, I was able to visualize what needed to be completed and all that we necessarily needed was a type of switch statement converted from C++ into Verilog.

3. Describe problems you faced and how you solved them:

Fortunately, this lab was straight-forward enough to follow the directions and not have any issues. Once I had visualized and mapped out the logic necessary for the project, the code fell into place and we were able to demonstrate the concept fairly easily.

4. Do you verify that the code included with this report is yours and your partner's original work (yes/no)?

Yes - our code is inherently original. However, we've helped several groups that were having trouble in our lab period (as usual) so there may be similarities in logic and/or syntax.

Implemented Schematic

N/A (Not necessary in this lab as Part A & B were neglected by the TA)

Verilog Code Implemented

```
module laser_surgery_sys #( parameter NBITS = 32 )(
input wire b ,   // button
input wire clk , // clock
output reg light // light register
);// End of module ---------------------------------

// --------------------------------------
// Comb. Logic - FSM
// --------------------------------------

wire timer;
reg reset_count;

reg [1:0] current_state = 2'b00;
reg [1:0] next_state = 2'b00;

// Initializing bits for behavioral logic . . .
localparam OFF = 2'b00;
localparam ON = 2'b10;
localparam START = 2'b01;

// Size of register (32) in form [n-1:0] where 32 is n . . .
reg [31:0] cnt_ini = 32'd0;
reg [31:0] cnt_rst = 32'd500000000;

always @(posedge clk) begin
        current_state <= next_state;
end
always @( current_state or b or timer) begin
// Switch case similar to C++ syntax ------------
        case(current_state)
                OFF : begin // OFF ---------------
                        next_state = OFF;         // Inf. Loop of OFF
                        reset_count = 1;          // " "
                        light = 1'b0;             // Initializes light
                        if(b) next_state = START; // If button is pressed . . .
                        end

                START : begin // START -----------
```

```
                        reset_count = 0; // Reinitializes count
                        light = 1'b0;    // Reinitializes light
                        next_state = ON; // Forces next state to be ON
                        end

                ON : begin // ON -----------------
                        next_state = ON; // Forces next state to be ON
                        reset_count = 0; // Reinitializes count
                        light = 1'b1;    // Reinitializes light
                        next_state = timer ? OFF : ON; // Ternary op for
                                                       determining the next
                                                       state
                        end

                default : begin // DEFAULT CASE -------
                        reset_count = 1'b1; // Reinitializes count
                        light = 1'b0;       // Reinitializes light
                        next_state = OFF;   // Forces next state to be OFF
                        end
        endcase // End of switch statement ------

end // End of local scope ---


// --------------------------------------
// Timer instantiation
// --------------------------------------
timer_st #( .NBITS(NBITS) ) timerst (
.timer(timer),
.clk(clk),
.reset(reset) ,
.cnt_ini(cnt_ini) ,
.cnt_rst(cnt_rst)
);
endmodule

module flopr #( parameter NBITS = 16 )(
 input clk,
input reset,
input [NBITS-1:0] cnt_ini,
input [NBITS-1:0] nextq,
output[NBITS-1:0] q
);
reg [NBITS-1:0] iq ;
always @(posedge clk) begin
```

```
 if (reset) begin
iq <= cnt_ini ;
 end
 else begin
iq <= nextq;
 end
end
assign q = iq ;
endmodule

module comparatorgen_st #( parameter NBITS = 16 )(
output wire r ,
input wire[NBITS-1:0] a ,
input wire[NBITS-1:0] b );
wire [NBITS-1:0] iresult ;
genvar k ;
generate
for (k=0; k < NBITS; k = k+1)
begin : blk
 xor c1 (iresult[k], a[k], b[k] ) ;
 end
endgenerate
// Reduction plus negation
assign r = ~(|iresult);
endmodule


module fulladder_st(
output wire r,
output wire cout,
input wire a,
input wire b,
input wire cin
) ;
assign r = (a ^ b) ^ (cin) ;
assign cout = (a & b) | ( a & cin ) | ( b & cin ) ;
endmodule


module addergen_st #( parameter NBITS = 16 )(
output wire[NBITS-1:0] r ,
output wire cout ,
input wire[NBITS-1:0] a ,
input wire[NBITS-1:0] b ,
input wire cin ) ;
```

```
wire [NBITS:0] carry;
assign carry[0]= cin ;
genvar k ;
generate
for (k=0; k < NBITS; k = k+1)
begin : blk
fulladder_st FA (
 .r(r[k]),
 .cout(carry[k+1]),
 .a(a[k]),
 .b(b[k]),
 .cin(carry[k]) ) ;
end
endgenerate
assign cout = carry[NBITS] ;
endmodule


module adder #( parameter NBITS = 16 )(
input [NBITS-1:0] q ,
input [NBITS-1:0] cnt_ini ,
input [NBITS-1:0] cnt_rst ,
output[NBITS-1:0] nextq,
output tick
);
wire same ;
wire[NBITS-1:0] inextq;
// ------------------------------------------------
// inextq = q + 1 ;
// ------------------------------------------------
addergen_st #(.NBITS(NBITS))
nextval ( .r(inextq), // Next value
 .cout(), // Carry out - Don't use
 .a(q), // Current value
 .b(16'b0000_0001), // Plus One
 .cin(16'b0000_0000) ) ; // No carry in
// ------------------------------------------------
// Are inextq and cnt_rst equal ?
// ------------------------------------------------
comparatorgen_st #(.NBITS(NBITS))
comparator (
.r(same) ,
.a(inextq),
.b(cnt_rst) );
```

```
// ----------------------------------------------------------------
// If they are the same produce a tick and set the value for nextq
// ----------------------------------------------------------------
assign tick = (same) ? 'd1 : 'd0 ;
assign nextq = (same) ? cnt_ini : inextq ;
endmodule


module timer_st #(
 parameter NBITS = 32
)
(
 output wire timer ,
 input wire clk ,
 input wire reset,
 input [NBITS-1:0] cnt_ini ,
 input [NBITS-1:0] cnt_rst
);
wire [NBITS-1:0] q ;
wire [NBITS-1:0] qnext ;
// Compute the next value
adder #( .NBITS(NBITS ) )
c1 (.q(q),
.cnt_ini(cnt_ini),
.cnt_rst(cnt_rst),
.nextq(qnext),
.tick(timer) );
// Save the next state
flopr #( .NBITS(NBITS ) )
 c2 (.clk(clk),
.reset(reset),
.cnt_ini(cnt_ini),
.nextq(qnext),
.q(q) );
endmodule
```

UCF File

```
// Inputs ------
NET “clk” LOC = “B8” ;
NET “b”   LOC = “G12” ;

// Output ------
NET “light” LOC = “M5” ;
```

Conclusion

In this lab, we timed the clock on the BASYS FPGA board and it remained lit for XXXXXX seconds. It is concluded that this implementation of our FSM (Finite State Machine), it is dependent on time with respect to waiting a certain amount of time while performing a particular task. In this case, it was illuminating the LED on our BASYS FPGA board.

As a whole, the circuit design and Boolean logic allowed us to see the implementation of the adder in Verilog and thus onto our FPGA BASYS board. Once the results expected were yielded, our component system was successfully completed.