Name: **Jonathan Lemarroy**

ECE 3544: Digital Design 1
Homework Assignment 6 (100 points)

Problem 1)

Problem 2)

```verilog
module problem2_jdl25175(clock, reset_l, count, state, carry);
   input        clock;      // System clock
   input        reset_l;    // Asynchronous active-low reset
   input        count;      // Synchronous active-high count enable
   output [3:0] state;      // Counter state
   output       carry;      // Counter carry-out
   reg    [3:0] state;

    always @(posedge clock or negedge reset_l) begin
        if(reset_l == 1'b0)
            state <= 4'b0000;
        else if(count) begin
            if(state == 4'b1001)
                state <= 4'b0000;
            else
                state <= state + 1'b1;
        else
            state <= state;
        end
    end
    assign carry = (state == 4'b1001) && count;
endmodule
```

Problem 3)

```verilog
module problem3_jdl25175(clock, reset_l, load, count, ins, state, carry);
   input        clock;      // System clock
   input        reset_l;    // Asynchronous active-low reset
   input        load;       // Synchronous active-high load enable
   input        count       // Synchronous active-high count enable
   input  [3:0] ins;        // Parallel load inputs
   output [3:0] state;      // Counter state
   output       carry;      // Counter carry-out
   reg    [3:0] state;

   always @(posedge clock or negedge reset_l) begin
        if(reset_l == 1'b0)
            state <= 4'b0000;
        else if(load & ~count)
            state <= ins;
        else if(count & ~load)
            state <= state + 1;
        else
            state <= state;
   end

   assign carry = count & (state == 4'b1111);

endmodule
```

Problem 4)

```
module problem4_jdl25175(clock, reset_l, count, state);
   input        clock;      // System clock;
   input        reset_l;    // Asynchronous active-low reset;
   input        count;      // Synchronous active-high count enable
   output [3:0] state;      // Counter state

   wire         carry;

   problem3_jdl25175 mod1(clock, reset_l, 1'b0, count, 4'b0000, state, carry);

endmodule
```

Problem 5)

```
module problem5_jdl25175(clock, clear, load, enable, updn, ins, count, carry);
    input        clock; // System clock
    input        clear; // SYNCHRONOUS ACTIVE-HIGH clear
    input        load;  // Synchronous active-high load enable
    input        enable;// Synchronous active-high count enable
    input        updn;  // Synchronous up-down control
    input  [3:0] ins;   // Parallel load inputs
    output [3:0] count; // Counter state
    output       carry; // Counter carry-out
    reg    [3:0] count;

    always @(posedge clock)
        if(clear)
            count = 4'b0000;
        else if(load)
            count = ins;
        else if(enable & ~updn)
            count = count + 4'b0001;
        else if(enable & updn)
            count = count = 4'b0001;
        else
            count = count;

    assign carry = ((count == 4'b1111) && enable && ~updn) || ((count == 4'b0000) && enable &
& updn);

endmodule
```

Problem 6)

```
module problem6_jdl25175(clock, reset_l, state);
   input        clock;
   input        reset_l;        // SYNCHRONOUS ACTIVE-LOW RESET;
   output [3:0] state;

   reg clr, load, updn;
   reg [3:0] ins;
   wire      carry;

   problem5_jdl25175 mod1(clock, clr, load, 1'b1, updn, ins, state, carry);

   always @(posedge clock) begin
        if(reset_l == 1'b0)
            clr = 1'b1;
        else begin
            clr = 1'b0;
            if(state == 4'b0111) begin
                updn = 1'b1;
                load = 1'b1;
                ins = 4'b1111;
            end
            else if(state == 4'b1000) begin
                updn = 1'b0;
                load = 1'b1;
                ins = 4'b0000;
            end
            else
                load = 1'b0;
        end
   end

endmodule
```

Problem 7)

```
module problem7_jdl25175(clock, reset_l, enable, state, out);
   input         clock;      // System clock
   input         reset_l;    // Asynchronous active-low reset
   input         enable;     // Synchronous active-high count enable
   output [25:0] state;      // Counter state.
   output        out;        // Output pulse.

   reg   [25:0] state;
   reg          out;

    always @(posedge clock, negedge reset_l) begin
        if(reset_l == 1'b0)
            state <= 26'b0;
        else if(enable) begin
            if(state == 26'b0) begin
                out <= 1'b1;
                state <= state + 26'b1;
            end
            else if(state == 26'd49999999) begin
                out <= 1'b0;
                state <= 26'b0;
            end
            else begin
                out <= 1'b0;
                state <= state + 26'b1;
            end
        end
        else begin
            state <= 26'b0;
        end
    end

endmodule
```

Problem 8)
The transmitter has a required propagation delay of 15ns + 18ns + 20ns = 53ns, while the receiver has a required delay of 18ns + 6ns + 20ns = 44ns. This makes the minimum period of the clock to be **53ns**, however, it should be more than that to be safe.

Problem 9)
I do not own the textbook to look up this problem.