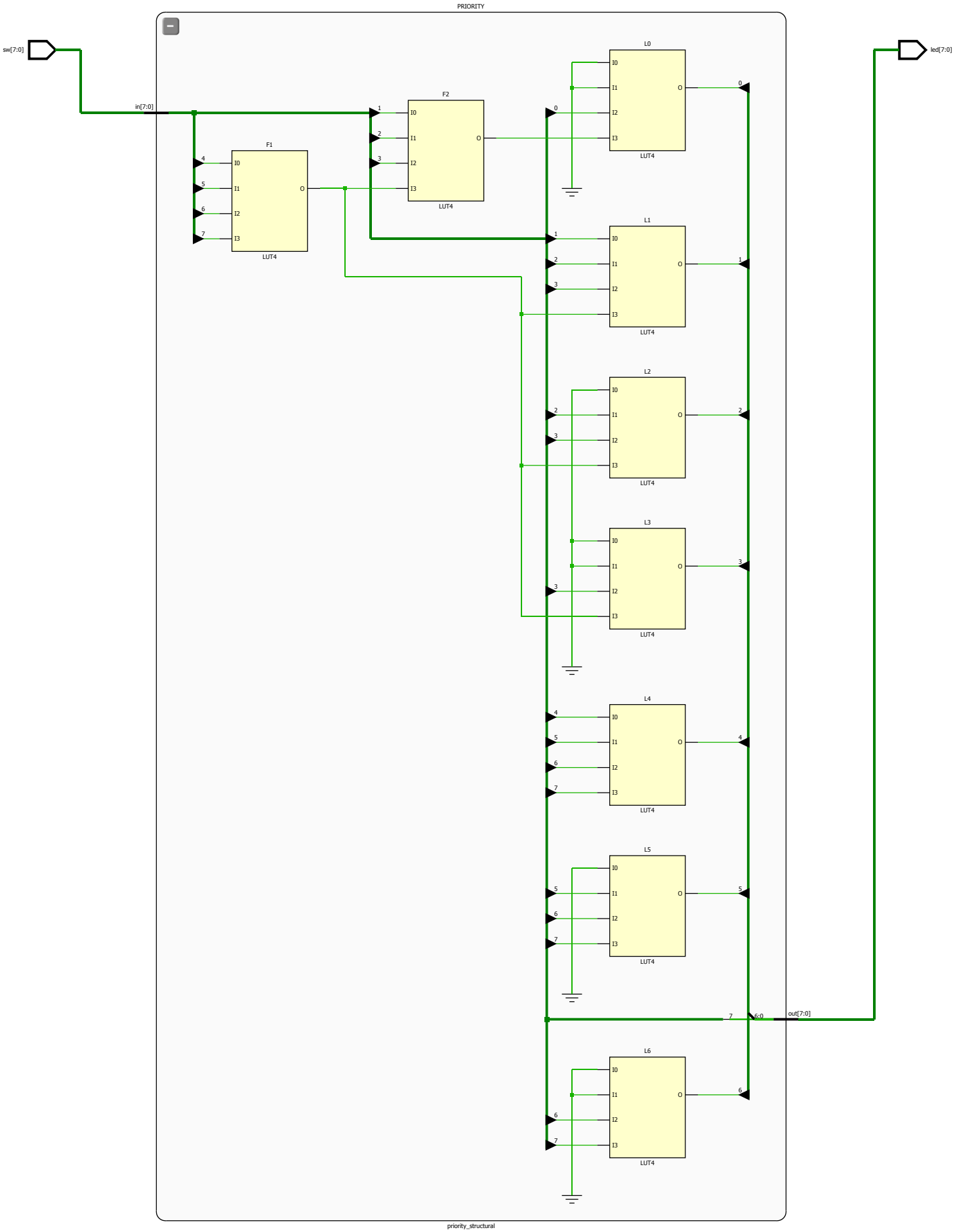
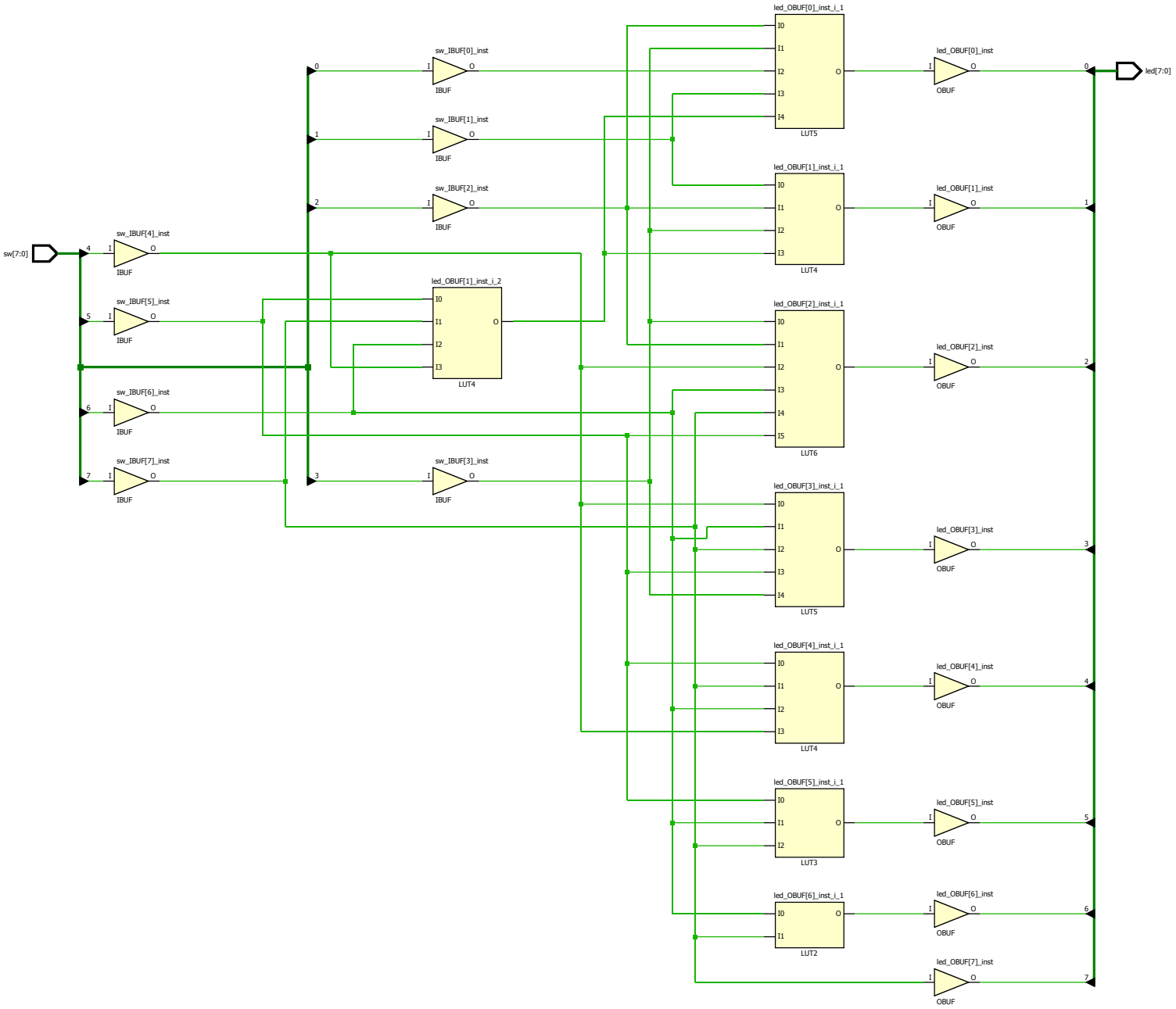
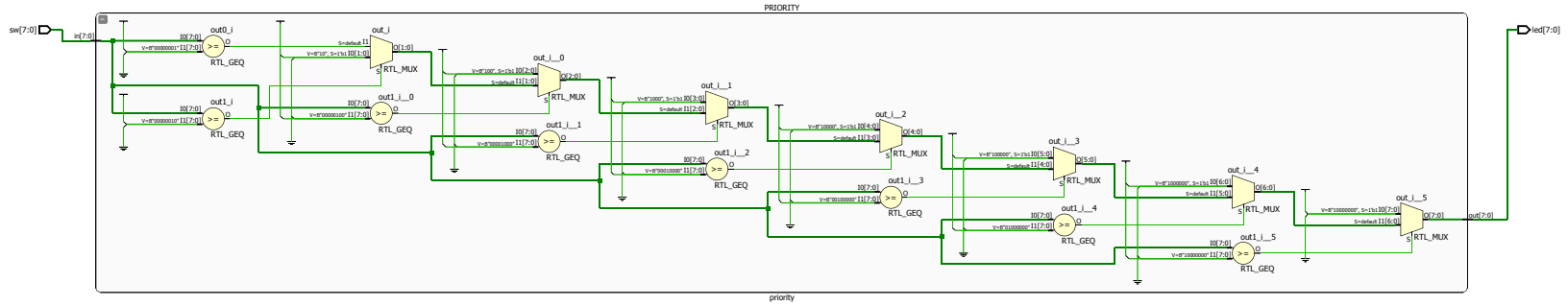
```
// File: priority_circuit.v
// John Hubbard, 16 Feb 205
// For HW #4 assignment of FPGA class.

//
// Answers to problem 3.9.4
//
// This is called a priority circuit. That is because higher priority inputs
// override lower priority inputs.

module priority_structural(in, out);
    input [7:0]in;
    output [7:0] out;

    wire f1, f2;

    buf B1(out[7], in[7]);
    LUT4 #(16'h00F0) L6(out[6], 'bx,   'bx,   in[6], in[7]);
    LUT4 #(16'h000C) L5(out[5], 'bx,   in[5], in[6], in[7]);
    LUT4 #(16'h0002) L4(out[4], in[4], in[5], in[6], in[7]);

    LUT4 #(16'h0001) F1(f1, in[4], in[5], in[6], in[7]);
    LUT4 #(16'h0100) F2(f2, in[1], in[2], in[3], f1);

    LUT4 #(16'hF000) L3(out[3], 'bx,   'bx,   in[3], f1);
    LUT4 #(16'h0C00) L2(out[2], 'bx,   in[2], in[3], f1);
    LUT4 #(16'h0200) L1(out[1], in[1], in[2], in[3], f1);

    LUT4 #(16'hF000) L0(out[0], 'bx, 'bx,     in[0], f2);

endmodule

module priority_behavioral(in, out);
    input [7:0]in;
    output [7:0] out;

    reg [7:0]out;

    always @(*)
    begin
        if (in >= (1 << 7))
            out = 8'b10000000;
        else if (in >= (1 << 6))
            out = 8'b01000000;
        else if (in >= (1 << 5))
            out = 8'b00100000;
        else if (in >= (1 << 4))
            out = 8'b00010000;
        else if (in >= (1 << 3))
            out = 8'b00001000;
        else if (in >= (1 << 2))
            out = 8'b00000100;
        else if (in >= (1 << 1))
            out = 8'b00000010;
```

```
        else if (in >= (1 << 0))
            out = 8'b00000001;
        else
            out = 8'b00000000;
    end
endmodule


module priority_top_module(sw, led);
    input  [7:0] sw; /* 76543210 */
    output [7:0] led; /* 76543210 */

//  priority_behavioral PRIORITY(.in(sw), .out(led));
    priority_structural PRIORITY(.in(sw), .out(led));
endmodule
```


PRIORITY
sw[7:0]
led[7:0]
in[7:0]
F1
F2
L0
L1
L2
L3
L4
L5
L6
LUT4
I0
I1
I2
I3
O
out[7:0]
6:0
priority_structural


sw[7:0]
led[7:0]
sw_IBUF[0]_inst
sw_IBUF[1]_inst
sw_IBUF[2]_inst
sw_IBUF[3]_inst
sw_IBUF[4]_inst
sw_IBUF[5]_inst
sw_IBUF[6]_inst
sw_IBUF[7]_inst
IBUF
led_OBUF[0]_inst_i_1
led_OBUF[1]_inst_i_1
led_OBUF[1]_inst_i_2
led_OBUF[2]_inst_i_1
led_OBUF[3]_inst_i_1
led_OBUF[4]_inst_i_1
led_OBUF[5]_inst_i_1
led_OBUF[6]_inst_i_1
LUT5
LUT4
LUT6
LUT3
LUT2
led_OBUF[0]_inst
led_OBUF[1]_inst
led_OBUF[2]_inst
led_OBUF[3]_inst
led_OBUF[4]_inst
led_OBUF[5]_inst
led_OBUF[6]_inst
led_OBUF[7]_inst
OBUF


PRIORITY
sw[7:0]
in[7:0]
led[7:0]
out[7:0]
out0_i
out1_i
out_i
out1_i__0
out_i__0
out1_i__1
out_i__1
out1_i__2
out_i__2
out1_i__3
out_i__3
out1_i__4
out_i__4
out1_i__5
out_i__5
RTL_GEQ
RTL_MUX
priority

UCSC 30207 - FPGA

John Hubbard

Hw #4

Problem 3.9.4 - Testing sw, fobes and LEDs.

1) This is called a priority circuit.

3) Using behavioral Verilog, 8 LUTs are required. However, it uses LUT5 and LUT6. So it has an advantage, due to not being constrained to LUT4.

4) Using structural Verilog, 9 LUT4's are required. It's hard to compare directly, but this is at least as good as the behavioral Verilog result and the tool-generated circuit.

```
// File: all_patterns.v
// John Hubbard, 16 Feb 205
// For HW #4 assignment of FPGA class.

//
// Answers to problem 3.9.7
//
// There are 2**7, or 128 possible patterns for a 7-seqment LED. It will take
// 128 seconds, or 2 minutes and 8 seconds, to display them all.

module all_patterns_top_module(clk, btnU, seg, an);
    parameter C = 28; //27..0 counter
    parameter N7 = 7 ;
    parameter N4 = 4 ;
    parameter CRYSTAL = 100 ; // 100 MHZ
    parameter NUM_SEC = 1 ;
    parameter STOPAT = (CRYSTAL * 1_000_000 * NUM_SEC)- 1 ;

    input clk, btnU;
    output [N7-1:0] seg; // These are backwards, to match the zero_to_127_counter
    wire   [N7-1:0] zero_to_127_counter;
    output [N4-1:0] an;
    wire [C-1:0] big_counter;
    wire one_second_clock;

    assign an = 4'b0111; // ON off off off
    mod_counter #(C,STOPAT) MOD_COUNTER(clk, btnU, big_counter, one_second_clock);
    counter #(N7) COUNTER(clk, btnU, one_second_clock, zero_to_127_counter);
    assign seg = zero_to_127_counter;
endmodule

module counter(clk, arst, en, q) ;
    parameter N = 7 ;
    input clk,arst,en ;
    output [N-1:0] q ;
    reg [N-1:0] q ;

    always @(posedge clk or posedge arst)
    if (arst == 1'b1)
        q <= 0 ;
    else if (en)
        q <= q + 1 ;
endmodule

module mod_counter(clk, arst, q, done) ;
    parameter N = 7 ;
    parameter MAX = 127 ;
    input clk,arst;
    output [N-1:0] q ;
    output done ;

    reg [N-1:0] q ;
    reg done ;
```

```
    always @(posedge clk or posedge arst)
    begin
        if (arst == 1'b1)
        begin
            q <= 0 ;
            done <= 0 ;
        end
        else if (q == MAX)
        begin
            q <= 0 ;
            done <= 1 ;
        end
        else
        begin
            q <= q + 1 ;
            done <= 0 ;
        end
    end
endmodule
```

```
// File: heartbeat.v
// John Hubbard, 16 Feb 205
// For HW #4 assignment of FPGA class.

//
// Heart beat display: problem 3.9.8
//

module choose_pattern_from_count(count, seg, an);
    input [2:0] count;
    output [6:0] seg; // abcdefg
    output [0:3] an;

    reg [6:0] seg;
    reg [0:3] an;

    always @(*)
        case (count)
            0:
            begin
                seg = 7'b0011100; // abgf
                an  = 4'b0111;    // leftmost
            end
            1:
            begin
                seg = 7'b0011100; // abgf
                an  = 4'b1011;    // left center
            end
            2:
            begin
                seg = 7'b0011100; // abgf
                an  = 4'b1101;    // right center
            end
            3:
            begin
                seg = 7'b0011100; // abgf
                an  = 4'b1110;    // rightmost
            end
            4:
            begin
                seg = 7'b1100010; // cdeg
                an  = 4'b1110;    // rightmost
            end
            5:
            begin
                seg = 7'b1100010; // cdeg
                an  = 4'b1101;    // right center
            end
            6:
            begin
                seg = 7'b1100010; // cdeg
                an  = 4'b1011;    // left center
            end
            7:
```

```
            begin
                seg = 7'b1100010; // cdeg
                an  = 4'b0111;    // leftmost
            end
        endcase

endmodule

module heartbreat_top_module(clk, btnU, seg, an);
    parameter C = 28; //27..0 counter
    parameter N7 = 7 ;
    parameter N4 = 4 ;
    parameter CRYSTAL = 100 ; // 100 MHZ
    parameter NUM_SEC = 1 ;
    parameter STOPAT = (CRYSTAL * 1_000_000 * NUM_SEC)- 1 ;

    input clk, btnU;
    output [0:6] seg; // abcdefg
    wire   [2:0] zero_to_8_counter;
    output [3:0] an;
    wire [C-1:0] big_counter;
    wire one_second_clock;

    mod_counter #(C,STOPAT) MOD_COUNTER(clk, btnU, big_counter, one_second_clock);
    counter #(3) COUNTER(clk, btnU, one_second_clock, zero_to_8_counter);

    choose_pattern_from_count PATTERN(zero_to_8_counter, seg, an);
endmodule
```

```
// File: random_numbers.v
// John Hubbard, 16 Feb 205
// For HW #4 assignment of FPGA class.

//
// Random numbers display: problem 4.7.2
//

module rom(in,out) ;
    parameter N = 3 ;
    parameter O = 14 ; //9999
    input [N-1:0] in ;
    output reg [O-1:0] out ;

    always @(in)
    begin
        case (in)
            0: out = 1;
            1: out = 17 ;
            2: out = 23 ;
            3: out = 57 ;
            4: out = 234 ;
            5: out = 9 ;
            6: out = 4878 ;
            7: out = 9999 ;
        default: out = 9998 ;
        endcase
    end
endmodule

module random_numbers_top_module(clk, btnU, seg, an);
    parameter C = 28; //27..0 counter
    parameter N7 = 7 ;
    parameter N4 = 4 ;
    parameter CRYSTAL = 100 ; // 100 MHZ
    parameter NUM_SEC = 1 ;
    parameter STOPAT = (CRYSTAL * 1_000_000 * NUM_SEC)- 1 ;

    wire [13:0] number;
    wire [15:0] text;

    input clk, btnU;
    output [0:6] seg; // abcdefg
    wire   [2:0] zero_to_8_counter;
    output [3:0] an;
    wire [C-1:0] big_counter;
    wire one_second_clock;

    mod_counter #(C,STOPAT) MOD_COUNTER(clk, btnU, big_counter, one_second_clock);
    counter #(3) COUNTER(clk, btnU, one_second_clock, zero_to_8_counter);

    rom THE_ROM(zero_to_8_counter, number);

    binary2bcd BCD(number, text);
```

```
    // TODO: this needs to handle four digit display
    display DISPLAY(text, clk, btnU, seg, an);

endmodule
```

```
//  File: binary2bcd.v
// John Hubbard

`timescale 1ns/1ns

module binary2bcd(number, text);
    parameter N = 16 ;
    parameter W = 4 ;
    input [N-1:0] number ;
    reg [N-1:0] workingNumber ;
    output reg [N-1:0] text ;

    reg [3:0] digit;
    integer digitEnd;

    always @(number)
    begin
        text          = 16'h0000;
        workingNumber = number;
        digit         = workingNumber % 10;
        digitEnd      = 3;

        while(workingNumber > 0 && (digitEnd < N))
        begin
            text[digitEnd-:4] = digit;

            // Move to the next digit:
            workingNumber = workingNumber / 10;
            digit = workingNumber % 10;
            digitEnd = digitEnd + 4;
        end
    end
endmodule

module binary2bcd_tb();
    parameter N = 16 ;
    reg [N-1:0] vec ;
    wire [N-1:0] text ;

    binary2bcd BCD(vec, text);

    initial
    begin
        for (vec = 0; vec < 102; vec = vec + 1)
        begin
            #1 $display("vec: %d, text: %h", vec, text);
        end
    end

endmodule
```

```
// File: display.v
// John Hubbard, 16 Feb 2015

module simple_counter(clk, arst,q) ;
    parameter N = 7 ;
    input clk,arst;
    output reg [N-1:0] q ;

    always @(posedge clk or posedge arst)
        if (arst == 1'b1)
            q <= 0 ;
        else
            q <= q + 1 ;
endmodule

module decoder(text,s,y,val) ;
    input [15:0] text;
    input [1:0] s ;
    output reg [3:0] y ;
    output reg [3:0] val ;

    always @(*)
    begin
        case (s)
            0: begin
                y <= 4'b1110 ;
                val <= text[3:0] ;
            end
            1:begin
                y <= 4'b1101 ;
                val <= text[7:4] ;
            end
            2:begin
                y <= 4'b1011 ;
                val <= text[11:8] ;
            end
            3: begin
                y <= 4'b0111 ;
                val <= text[15:12] ;
            end
            default:begin
                y <= 4'bx ;
                val <= 4'bx ;
            end
        endcase
    end
endmodule

module display(text,clk, arst, seg, an) ;
    parameter C = 21;
    parameter N7 = 7 ;
    parameter N4 = 4 ;
    parameter N2 = 2 ;
    parameter ANODE_FREQ = 20 ; //20 = 47 HZ; 19 is 95 HZ 95-47 = 48HZ
```

```
    input [15:0] text;
    input clk, arst;
    output [0:N7-1] seg ;
    output [N4-1:0] an ;

    wire [C-1:0] q ;
    wire [N2-1:0] sel ;
    wire [N4-1:0] zero_to_f_counter ;

    assign sel[1] = q[ANODE_FREQ] ;
    assign sel[0] = q[ANODE_FREQ-1] ;

    //assign sel[1] = q[15] ; 23 hz
    //assign sel[0] = q[20] ; 762 hz
    //Run with above number and see display
    simple_counter #C U(clk,arst,q) ;
    decoder D(text,sel,an,zero_to_f_counter) ;
    hex2_7seg H(zero_to_f_counter,seg);
endmodule

module display_test(clk, btnU, seg, an) ;
    parameter D = 'h8602 ;
    parameter N7 = 7 ;
    parameter N4 = 4 ;
    input clk, btnU;
    output [0:N7-1] seg ;
    output [N4-1:0] an ;

    wire [15:0] text;
    assign text = D ;
    display T(text,clk,btnU, seg, an) ;
endmodule
```