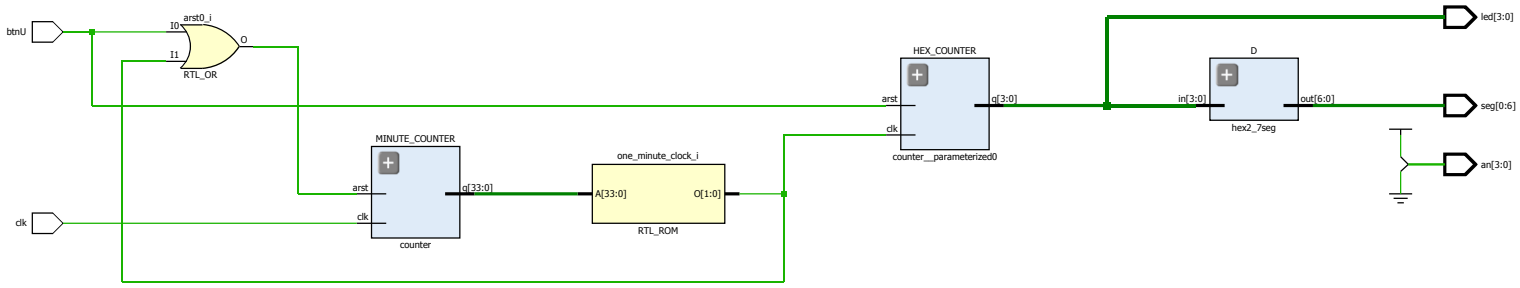
```
// File counter.v
// Also uses: hex2_7seg.v basys3.xdc
// Modified (and fixed) by John Hubbard, 06 Feb 2015
//
// Used integer and 34-bit wire variables, to hold the large counter values.
// When counting up to 60 seconds, using a 100 MHz clock, you need to count
// up to 60*100*1,000,000 = 6 billion, which requires at least 33 bits.
//
// The original code used "parameter" to hold this, but the parameter keyword
// is only necessary when a value needs to be changed at instantiation time.
// It appears that only 32 bits wide parameters are supported, which led to a
// bug that was invisible for a one-second (100 * 1,000,000 : 27 bits) counter,
// but overflowed a 32-bit counter.
//
// The fix, again, is simply to use wire and integer types of large enough bit
// width.
//
// --John Hubbard, 06 Feb 2015
//
// For HW #3 assignment of FPGA class.

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

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

module oneminute_bad(clk, btnU, seg, an, led) ;
    parameter C = 34 ; // 36 bits is ample to count up 60 * 100 MHz.
    parameter N7 = 7 ;
    parameter N4 = 4 ;

    integer CRYSTAL = 100 ; // 100 MHZ
    wire  [C-1:0]STOPAT_ONE_MIN = (CRYSTAL * 1_000_000 * 60)- 1;

    input clk, btnU;
    output [0:N7-1] seg ;
    output [N4-1:0] an ;
    output [N4-1:0] led ;
    wire [C-1:0] sec_counter ;
    wire [N4-1:0] zero_to_f_counter ;
    wire one_minute_clock ;

    assign an = 4'b0010 ; /* ON ON OFF ON */
    assign one_minute_clock = (sec_counter == STOPAT_ONE_MIN) ? 1: 0 ;

    counter #C MINUTE_COUNTER(clk, (btnU || one_minute_clock), sec_counter) ;
```

```
    counter #N4 HEX_COUNTER(one_minute_clock, btnU, zero_to_f_counter) ;

    hex2_7seg D(zero_to_f_counter,seg);

    assign led = zero_to_f_counter;
endmodule
```


btnU
arst0_i
I0
I1
O
RTL_OR
clk
MINUTE_COUNTER
arst
clk
q[33:0]
counter
one_minute_clock_i
A[33:0]
O[1:0]
RTL_ROM
HEX_COUNTER
arst
clk
q[3:0]
counter__parameterized0
D
in[3:0]
out[6:0]
hex2_7seg
led[3:0]
seg[0:6]
an[3:0]

```v
//file and2gate.v
module and2gate(a,b,f) ;
    input a,b ;
    output f ;
    assign f = a & b ;
endmodule

module a7_and2gate(sw, led) ;
    input [7:6] sw;
    output [4:4] led ;
    and2gate U1(sw[7],sw[6],led[4]) ;
endmodule
```

```
// file hex2_7seg.v
// Modified by John Hubbard, 31 Jan 2015
// Added hex2_7seg_structural module, and tested by instantiating into
// the switch-based test (and also the one-minute counter).
//
// For HW #3 assignment of FPGA class.

module hex2_7seg(in,out) ;
    input [3:0] in ; /*3210 */
    output [6:0] out ; /* abcdefg */
    reg [6:0] out ;

    always @(*)
    begin
        case (in)
             /*3210*/ /*abcdefg */
             0:  out = 7'b0000001 ;
             1:  out = 7'b1001111 ;
             2:  out = 7'b0010010 ;
             3:  out = 7'b0000110 ;
             4:  out = 7'b1001100 ;
             5:  out = 7'b0100100 ;
             6:  out = 7'b0100000 ;
             7:  out = 7'b0001111 ;
             8:  out = 7'b0000000 ;
             9:  out = 7'b0000100 ;
             10: out = 7'b0001000 ;
             11: out = 7'b1100000 ;
             12: out = 7'b0110001 ;
             13: out = 7'b1000010 ;
             14: out = 7'b0110000 ;
             15: out = 7'b0111000 ;
            default: out = 7'bx ;
         endcase
     end
endmodule

module hex2_7seg_structural(in,out) ;
    input [3:0] in ; /*3210 */
    output [6:0] out ; /* abcdefg */
    wire [6:0] out ;

    LUT4 #(16'h1083) L0(out[0], in[0], in[1], in[2], in[3]);
    LUT4 #(16'h208e) L1(out[1], in[0], in[1], in[2], in[3]);
    LUT4 #(16'h02ba) L2(out[2], in[0], in[1], in[2], in[3]);
    LUT4 #(16'h8492) L3(out[3], in[0], in[1], in[2], in[3]);
    LUT4 #(16'hd004) L4(out[4], in[0], in[1], in[2], in[3]);
    LUT4 #(16'hd860) L5(out[5], in[0], in[1], in[2], in[3]);
    LUT4 #(16'h2812) L6(out[6], in[0], in[1], in[2], in[3]);
endmodule

module a7_hex2_7seg(sw,seg,an) ;
    input [3:0] sw ; /* 3210 */
    output [0:6] seg; /* abcdefg */
```

```
    output [3:0] an ; /* 3210 */

    assign an = 4'b0010 ; /* ON ON OFF ON */
    // hex2_7seg U(.in(sw),.out(seg) );
    hex2_7seg_structural U(.in(sw),.out(seg) );
endmodule
```