```
`timescale 1ns/1ns

module binary2bcdBehavioral(number,text) ;
    parameter N = 16 ;
    parameter W = 4 ;
    input [N-1:0] number ;
    output [N-1:0] text ;

    reg [N-1 :0] workingNumber ;
    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 add3(a, sum);
    input [0:3]a;
    output [0:3]sum;

    reg [3:0] sum;
    always @(a)
        case (a)
            4'b0000: sum <= 4'b0000;
            4'b1000: sum <= 4'b1000;
            4'b0100: sum <= 4'b0100;
            4'b1100: sum <= 4'b1100;
            4'b0010: sum <= 4'b0010;
            4'b1010: sum <= 4'b0001;
            4'b0110: sum <= 4'b1001;
            4'b1110: sum <= 4'b0101;
            4'b0001: sum <= 4'b1101;
            4'b1001: sum <= 4'b0011;
            default: sum <= 4'b0000;
        endcase

//  LUT4 #(16'h03E0) S3(sum[3], a[0], a[1], a[2], a[3]);
```

```
//  LUT4 #(16'h0210) S2(sum[2], a[0], a[1], a[2], a[3]);
//  LUT4 #(16'h018C) S1(sum[1], a[0], a[1], a[2], a[3]);
//  LUT4 #(16'h014A) S0(sum[0], a[0], a[1], a[2], a[3]);
endmodule

module add3_tb();
    reg [16:0] vec;
    wire [16:0] sum;

    add3 ADD3(vec, sum);

    initial
    begin
        for (vec = 0; vec < 16; vec = vec + 1)
        begin
            #1;
            $display("vec: %d, sum: %h", vec, sum);
        end
    end
endmodule

module binary2bcdStructral(n,text) ;
    parameter N = 16 ;
    parameter W = 4 ;
    input [N-1:0] n ;
    output [N-1:0] text ;

    wire [0:3] c1_out, c2_out, c3_out, c4_out, c5_out;

    buf BUF0(text[0], n[0]);
    buf BUF1(text[10], 1'b0);
    buf BUF2(text[11], 1'b0);

    add3 C1({n[5],      n[6], n[7], 1'b0}, c1_out);
    add3 C2({n[4],      c1_out[0:2]}, c2_out);
    add3 C3({n[3],      c2_out[0:2]}, c3_out);
    add3 C4({c3_out[3], c2_out[3], c1_out[3], 1'b0}, {c4_out[0:2], text[9]});
    add3 C5({n[2],      c3_out[0:2]}, c5_out);
    add3 C6({c5_out[3], c4_out[0:2]}, {text[5], text[6], text[7], text[8]});
    add3 C7({n[1],      c5_out[0:2]}, {text[1], text[2], text[3], text[4]});
endmodule

module d10(n,q,r);
    parameter K = 32 ;
    input [K-1:0] n ;
    output [K-1:0] q ;
    output [3:0] r ;

    assign q = n / 10;
    assign r = n % 10;
endmodule

module binary2bcdDivision(number,text) ;
    parameter N = 16 ;
```

```
    parameter W = 4 ;
    input [N-1:0] number ;
    output [N-1:0] text ;
    wire [N-1:0] q1, q2, q3, q4;
    wire [3:0]   r1, r2, r3, r4;

    d10 #(N) U1(number, q1, r1);
    d10 #(N) U2(q1, q2, r2);
    d10 #(N) U3(q2, q3, r3);
    d10 #(N) U4(q3, q4, r4);
    assign text = {r4, r3, r2, r1};
endmodule

module binary2bcd_tb();
    parameter N = 16 ;
    reg [N-1:0] vec ;
    wire [N-1:0] text ;
    wire [N-1:0] text1 ;
    wire [N-1:0] text2 ;
    wire [N-1:0] text3 ;

    binary2bcdDivision DIVISION_BY_10(vec, text);
    binary2bcdBehavioral BEHAVIORAL(vec, text1);
    binary2bcdStructral STRUCTURAL(vec, text2);

    initial
    begin
        for (vec = 0; vec < 1000; vec = vec + 1)
        begin
            #1 $display("vec: %d, text: %h, text1: %h, text2: %h",
                        vec, text, text1, text2);
        end
    end
endmodule
```

```
//  File: binary2bcd_test.v
// John Hubbard

// YOU CANNOT CHANGE ANYTHING BELOW
module binary2bcdTest(clk, btnU, seg, an,led) ;
    parameter N = 7 ;
    parameter W = 4 ;

    input clk, btnU ;
    output [0:N-1] seg ;
    output [W-1:0] an ;
    output[15:0] led ;

    wire done ;
    reg[15:0] number ;
    wire[15:0] text1,text2,text3,text ;

    always @(posedge clk or posedge btnU)
    begin
    if (btnU)
        number = 0;
    else if (done)
        number = number + 1 ;
    end

    binary2bcdBehavioral U1(number,text1) ;
    binary2bcdStructral U2(number,text2) ;
    binary2bcdDivision U3(number,text3) ;

    assign equal = ((text1 == text2) && (text1 == text3) && (text2 == text3))?1'b1:1'b0 ;
    assign text = {15'b0,equal} ;

    display_at_n_second #(0.5) D (clk, btnU, seg, an, text,done) ;

    assign led = text1 ;
endmodule
```

```
//Copyright: Jagadeesh Vasudevamurthy
//File seg7.v display_at_n_second.v

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

module display_at_n_second(clk, arst, seg, an, text, done) ;
    parameter NUM_SEC = 1 ;
    parameter C = 35; //27 for 1 sec
    parameter N = 7 ;
    parameter W = 4 ;
    parameter CRYSTAL = 100 ; // 100 MHZ
    parameter [C-1:0] STOPAT = (CRYSTAL * 1_000_000 * NUM_SEC)- 1 ;

    input clk, arst ;
    input [15:0] text ;
    output [0:N-1] seg ;
    output [W-1:0] an ;
    output done ;
    wire [C-1:0] clock ;

    mod_counter #(C,STOPAT) U(clk, arst, clock, done) ;
    display #(C)T(text,clk,arst, seg, an) ;
endmodule
```

```
// file hex2_7seg.v
// Modified by John Hubbard, 31 Jan 2015
// For HW #4 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
```

```
// 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
```