```
//----------------------------------------------------------------------
----
//-- CPE 133 Verilog File: sseg_dec.v
//-- Description: Special seven segment display driver;
//--
//--  two special inputs:
//--
//--      VALID: if valid = 0, four dashes will be display
//--             if valid = 1, decimal number appears on display
//--
//--       SIGN: if sign = 1, a minus sign appears in left-most digit
//--             if sign = 0, no minus sign appears
//--
//--
//-- revisions:
//----------------------------------------------------------------------
----

//-----------------------------------------------------------------
//-- 4 digit seven-segment display driver. Outputs are active
//-- low and configured ABCEDFG in "segment" output.
//-----------------------------------------------------------------
module sseg_dec( input [7:0] ALU_VAL,
                                  input SIGN,
                                  input VALID,
                 input CLK,
                 output [3:0] DISP_EN,
                 output [7:0] SEGMENTS);

   // intermediate signal declaration ----------------------
   reg   [1:0] cnt_dig;
   reg   [3:0] digit;
   wire   [3:0] lsd,msd,mmsd;
   reg [3:0] msd_v, mmsd_v;
   wire   sclk;

   // instantiation of bin to bcd converter -----------------
    bin2bcdconv my_conv( .BIN_CNT_IN(ALU_VAL),
                         .LSD_OUT(lsd),
                         .MSD_OUT(msd),
                         .MMSD_OUT(mmsd));

   clk_div my_clk( .clk(CLK),
                       .sclk(sclk) );

   // advance the count (used for display multiplexing) -----
   always @ (posedge sclk)
   begin
         cnt_dig <= cnt_dig + 1;
   end
```

```
 // select the display sseg data abcdefg (active low) -----
 assign SEGMENTS = (digit==0)? 8'b00000011:
             (digit==1)?8'b10011111:
             (digit==2)?8'b00100101:
             (digit==3)?8'b00001101:
             (digit==4)?8'b10011001:
             (digit==5)?8'b01001001:
             (digit==6)?8'b01000001:
             (digit==7)?8'b00011111:
             (digit==8)?8'b00000001:
             (digit==9)?8'b00001001:
                         (digit==4'hE)?8'b11111101:   // dash
                         (digit==4'hF)?8'b11111111:   // blank
             8'b11111111;

// actuate the correct display --------------------------
 assign DISP_EN = (cnt_dig==0)? 4'b1110:
                  (cnt_dig==1)? 4'b1101:
                  (cnt_dig==2)? 4'b1011:
                  (cnt_dig==3)? 4'b0111:
                                4'b1111;

      always@ (cnt_dig, lsd, msd, mmsd, SIGN, VALID)
      begin
          msd_v =msd;
          mmsd_v=mmsd;
               //do the lead zero blanking for two msb's
               if (mmsd == 4'h0)
               begin
                  if (msd == 4'h0)
                          msd_v = 4'hF;
                  mmsd_v = 4'hF;
               end;

               if (VALID == 1)
               begin
                       if (SIGN == 0)
                                case(cnt_dig)
                                    0: digit = 4'b1111;
                                         1: digit = mmsd_v;
                                         2: digit = msd_v;
                                         3: digit = lsd;
                                         default: digit = 4'b0000;
                                endcase
                       else
                                case(cnt_dig)
                  0: digit = 4'b1110;
                  1: digit = mmsd_v;
                  2: digit = msd_v;
```

```
                    3: digit = lsd;
                    default: digit = 4'b0000;
                endcase
                 end
                 else digit = 4'b1111;
        end

endmodule

//----------------------------------------------------------------------
//-- interface description for bin to bcd converter
//----------------------------------------------------------------------
module bin2bcdconv ( input [7:0] BIN_CNT_IN,
                     output reg [3:0] LSD_OUT,
                     output reg [3:0] MSD_OUT,
                     output reg [3:0] MMSD_OUT);
//----------------------------------------------------------------------
------------------
//-- description of 8-bit binary to 3-digit BCD converter using double
dabble algorithm
//----------------------------------------------------------------------
------------------
    integer i;
    always@(BIN_CNT_IN)
    begin
        LSD_OUT=0;
        MSD_OUT=0;
        MMSD_OUT=0;

        for(i=7; i>=0; i=i-1)
        begin
            //add 3 to columns >=5
            if(MMSD_OUT >=5)
                MMSD_OUT = MMSD_OUT +3;
            if(MSD_OUT >=5)
                MSD_OUT = MSD_OUT +3;
            if(LSD_OUT>=5)
                LSD_OUT = LSD_OUT +3;

            MMSD_OUT = MMSD_OUT <<1;
            MMSD_OUT[0] = MSD_OUT[3];
            MSD_OUT = MSD_OUT <<1;
            MSD_OUT[0] = LSD_OUT[3];
            LSD_OUT = LSD_OUT <<1;
            LSD_OUT[0] = BIN_CNT_IN[i];
        end
    end
endmodule
```

```
//----------------------------------------------------------------------
---
//-- Module to divide the clock
//----------------------------------------------------------------------
---
module clk_div (  input clk,
                  output sclk);

  integer MAX_COUNT = 2200;
  integer div_cnt =0;
  reg tmp_clk=0;

   always @ (posedge clk)
   begin
         if (div_cnt == MAX_COUNT)
         begin
            tmp_clk = ~tmp_clk;
            div_cnt = 0;
         end else
            div_cnt = div_cnt + 1;
   end
   assign sclk = tmp_clk;
endmodule
```