```
/*
   Ben Davis
   2/13/24
   EE 371
   Lab 4, Task 1

   This is a module that implements the fsm and asm modules
   to count how many 1's are in a byte. The byte is input
   through SW[7:0]. The start signal is wired to SW[9], the
   reset signal is wired to KEY[0]. When the counting is done
   LEDR[9] lights up, and HEX0 displays the number of ones.

*/

module lab4task1_fpga (
               input logic CLOCK_50, //clock pin
               input logic [3:0] KEY, //key0 for reset
               input logic [9:0] SW, //sw9 start, sw7-0 input data
               output logic [9:0] LEDR, //ledr9 is done signal
               output logic [6:0] HEX0); //displays number of ones

   //intermediate logic variables for asm and fsm
   logic done; //counting is done
   logic shr; //shift to the right
   logic st; //start counting
   logic [3:0] result; //the final result

   lab4task1_FSM fsm (.clk(CLOCK_50), .reset(~KEY[0]), .start(SW[9]),
                              .done(done), .s(st), .shr(shr));

   lab4task1_ASM asm (.clk(CLOCK_50), .in(SW[7:0]), .start(st),
                        .shr(shr), .zero(done), .result(result));

   //when done, turn on ledr9
   assign LEDR[9] = done;

   //hexzero is the input for HEX0, uses the input of result
   //to find the correct 7 bit output
   logic [6:0] hexzero;
   hexadecimal hx0 (.in(result), .out(hx0));

   assign HEX0 = hexzero;
endmodule
//testbench
module lab4task1_fpga_tb ();

   //recall variables
   logic CLOCK_50;
   logic [3:0] KEY;
   logic [9:0] SW;
   logic [9:0] LEDR;
   logic [6:0] HEX0;

   //reinstantiate module
   lab4task1_fpga dut (.CLOCK_50, .KEY, .SW, .LEDR, .HEX0);

   //clock setup
   parameter clk_pd = 100;
   initial begin
      CLOCK_50 <= 0;
      forever #(clk_pd /2) CLOCK_50 <= ~CLOCK_50;
   end //of setup for clock

   //tests an instance where reset is hit initially, and the
   //input is 01101100. start (SW[9]) is hit after one cycle
   initial begin
      KEY[0] <= 0; SW <= 10'b0001101100; @(posedge CLOCK_50);
      KEY[0] <= 1; SW <= 10'b1001101100; @(posedge CLOCK_50);
      KEY[0] <= 1; SW <= 10'b1001101100; @(posedge CLOCK_50);
      KEY[0] <= 1; SW <= 10'b1001101100; @(posedge CLOCK_50);
      KEY[0] <= 1; SW <= 10'b1001101100; @(posedge CLOCK_50);
      KEY[0] <= 1; SW <= 10'b1001101100; @(posedge CLOCK_50);
      KEY[0] <= 1; SW <= 10'b1001101100; @(posedge CLOCK_50);
```

```
        KEY[0] <= 1; SW <= 10'b1001101100; @(posedge CLOCK_50);
        KEY[0] <= 1; SW <= 10'b1001101100; @(posedge CLOCK_50);
        KEY[0] <= 1; SW <= 10'b1001101100; @(posedge CLOCK_50);
        KEY[0] <= 1; SW <= 10'b1001101100; @(posedge CLOCK_50);
        $stop;
    end
endmodule //testbench
```