```
/*
   Ben Davis
   2/15/24
   EE 371
   Lab 4, Task 2

   This module uses the fsm, datapath, and hex converting modules
   to search through a ram for a data input. CLOCK_50 pin is used,
   KEY[0] is the rest signal, SW9 is the start signal, SW7-0 are the
   data inputs. Its outputs are LEDR9 if the data is found, LEDR8 if
   the search is exhausted and the data is not found, and HEX0 and 1
   to display the data's address in the ram

*/

module binary_fpga (
                input logic CLOCK_50, //clock
                input logic [9:0] SW, //start and data input
                input logic [3:0] KEY, //reset
                output logic [9:0] LEDR, //found and not found
                output logic [6:0] HEX0, //data address first digit
                output logic [6:0] HEX1); //data address second digit
   //intermediate logic
   logic [4:0] L; //data address in ram
   logic clear, comp, hold, finish, found;

   fsm_bin fsm (.clk(CLOCK_50), .reset(~KEY[0]), .start(SW[9]), .finish(finish),
                .clear(clear), .comp(comp), .hold(hold));

   data_bin asm (.clk(CLOCK_50), .A(SW[7:0]), .clear(clear), .comp(comp), .hold(hold),
                .L(L), .finish(finish), .found(found));

   //logic for HEX0 display
   logic [6:0] hexd0;
   hexadecimal hx0 (.in(L[3:0]), .out(hexd0));

   //logic for HEX1 display
   logic [6:0] hexd1;
   hexadecimal hx1 (.in(L[4]), .out(hexd1));

   assign HEX0 = (found) ? hexd0 : 7'b1111111;
   assign HEX1 = (found) ? hexd1 : 7'b1111111;


endmodule
//testbench
`timescale 1ps/1ps
module binary_fpga_tb ();

   //reccall variables
   logic CLOCK_50;
   logic [9:0] SW, LEDR;
   logic [3:0] KEY;
   logic [6:0] HEX0, HEX1;

   binary_fpga dut (.CLOCK_50, .SW, .KEY, .LEDR, .HEX0, .HEX1);

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

   //tests an instance where the input is 8'b1, and start is
   //pressed after one clock cycle
   initial begin
      KEY <= 4'b1000; SW <= 10'b0000000001; @(posedge CLOCK_50);
      KEY <= 4'b1000; SW <= 10'b1000000001; @(posedge CLOCK_50);
      KEY <= 4'b1000; SW <= 10'b1000000001; @(posedge CLOCK_50);
      KEY <= 4'b1000; SW <= 10'b1000000001; @(posedge CLOCK_50);
      KEY <= 4'b1000; SW <= 10'b1000000001; @(posedge CLOCK_50);
      KEY <= 4'b1000; SW <= 10'b1000000001; @(posedge CLOCK_50);
```

```
74          KEY <= 4'b1000; SW <= 10'b1000000001; @(posedge CLOCK_50);
75          KEY <= 4'b1000; SW <= 10'b1000000001; @(posedge CLOCK_50);
76
77          $stop;
78      end
79  endmodule //test end
```