Name: Adithya M SRN: PES1UG20CS621 SEC: K

reg_file.v

```
// Write code for modules you need here
module reg16(input wire clk,reset,load,input wire [15:0]din,output wire
[15:0]dout);
      dfrl d0(clk,reset,load,din[0],dout[0]);
      dfrl d1(clk,reset,load,din[1],dout[1]);
      dfrl d2(clk,reset,load,din[2],dout[2]);
      dfrl d3(clk,reset,load,din[3],dout[3]);
      dfrl d4(clk,reset,load,din[4],dout[4]);
      dfrl d5(clk,reset,load,din[5],dout[5]);
      dfrl d6(clk,reset,load,din[6],dout[6]);
      dfrl d7(clk,reset,load,din[7],dout[7]);
      dfrl d8(clk,reset,load,din[8],dout[8]);
      dfrl d9(clk,reset,load,din[9],dout[9]);
      dfrl d10(clk,reset,load,din[10],dout[10]);
      dfrl d11(clk,reset,load,din[11],dout[11]);
      dfrl d12(clk,reset,load,din[12],dout[12]);
      dfrl d13(clk,reset,load,din[13],dout[13]);
      dfrl d14(clk,reset,load,din[14],dout[14]);
      dfrl d15(clk,reset,load,din[15],dout[15]);
endmodule

module mux8_16(input wire [15:0]i0,i1,i2,i3,i4,i5,i6,i7,input wire
j0,j1,j2,output wire [15:0]o);

      mux8 mx0({i0[0],i1[0],i2[0],i3[0],i4[0],i5[0],i6[0],i7[0]},j2,j1,j0,o[0]);
      mux8 mx1({i0[1],i1[1],i2[1],i3[1],i4[1],i5[1],i6[1],i7[1]},j2,j1,j0,o[1]);
      mux8 mx2({i0[2],i1[2],i2[2],i3[2],i4[2],i5[2],i6[2],i7[2]},j2,j1,j0,o[2]);
      mux8 mx3({i0[3],i1[3],i2[3],i3[3],i4[3],i5[3],i6[3],i7[3]},j2,j1,j0,o[3]);
      mux8 mx4({i0[4],i1[4],i2[4],i3[4],i4[4],i5[4],i6[4],i7[4]},j2,j1,j0,o[4]);
      mux8 mx5({i0[5],i1[5],i2[5],i3[5],i4[5],i5[5],i6[5],i7[5]},j2,j1,j0,o[5]);
      mux8 mx6({i0[6],i1[6],i2[6],i3[6],i4[6],i5[6],i6[6],i7[6]},j2,j1,j0,o[6]);
      mux8 mx7({i0[7],i1[7],i2[7],i3[7],i4[7],i5[7],i6[7],i7[7]},j2,j1,j0,o[7]);
      mux8 mx8({i0[8],i1[8],i2[8],i3[8],i4[8],i5[8],i6[8],i7[8]},j2,j1,j0,o[8]);
      mux8 mx9({i0[9],i1[9],i2[9],i3[9],i4[9],i5[9],i6[9],i7[9]},j2,j1,j0,o[9]);
      mux8
mx10({i0[10],i1[10],i2[10],i3[10],i4[10],i5[10],i6[10],i7[10]},j2,j1,j0,o[10]);
      mux8
mx11({i0[11],i1[11],i2[11],i3[11],i4[11],i5[11],i6[11],i7[11]},j2,j1,j0,o[11]);
      mux8
mx12({i0[12],i1[12],i2[12],i3[12],i4[12],i5[12],i6[12],i7[12]},j2,j1,j0,o[12]);
      mux8
mx13({i0[13],i1[13],i2[13],i3[13],i4[13],i5[13],i6[13],i7[13]},j2,j1,j0,o[13]);
      mux8
mx14({i0[14],i1[14],i2[14],i3[14],i4[14],i5[14],i6[14],i7[14]},j2,j1,j0,o[14]);
      mux8
mx15({i0[15],i1[15],i2[15],i3[15],i4[15],i5[15],i6[15],i7[15]},j2,j1,j0,o[15]);

endmodule


module reg_file (input wire clk, reset, wr, input wire [2:0] rd_addr_a,
rd_addr_b, wr_addr, input wire [15:0] d_in, output wire [15:0] d_out_a,
d_out_b);

// Declare wires here
```

```
      wire [0:7]load;
      wire [0:15]r0,r1,r2,r3,r4,r5,r6,r7;

// Instantiate modules here

      demux8 dmx(wr,wr_addr[2],wr_addr[1],wr_addr[0],load);

      reg16 reg0(clk,reset,load[0],d_in,r0);
      reg16 reg1(clk,reset,load[1],d_in,r1);
      reg16 reg2(clk,reset,load[2],d_in,r2);
      reg16 reg3(clk,reset,load[3],d_in,r3);
      reg16 reg4(clk,reset,load[4],d_in,r4);
      reg16 reg5(clk,reset,load[5],d_in,r5);
      reg16 reg6(clk,reset,load[6],d_in,r6);
      reg16 reg7(clk,reset,load[7],d_in,r7);

      mux8_16
mm0(r0,r1,r2,r3,r4,r5,r6,r7,rd_addr_a[0],rd_addr_a[1],rd_addr_a[2],d_out_a);
      mux8_16
mm1(r0,r1,r2,r3,r4,r5,r6,r7,rd_addr_b[0],rd_addr_b[1],rd_addr_b[2],d_out_b);

endmodule
```

tb_reg_file.v

```
`timescale 1 ns / 100 ps
`define TESTVECS 6

module tb;
  reg clk, reset, wr;
  reg [2:0] rd_addr_a, rd_addr_b, wr_addr; reg [15:0] d_in;
  wire [15:0] d_out_a, d_out_b;
  reg [25:0] test_vecs [0:(`TESTVECS-1)];
  integer i;
  initial begin $dumpfile("tb_reg_file.vcd"); $dumpvars(0,tb); end
  initial begin reset = 1'b1; #12.5 reset = 1'b0; end
  initial clk = 1'b0; always #5 clk =~ clk;
  initial begin
    test_vecs[0][25] = 1'b1; test_vecs[0][24:22] = 3'ox; test_vecs[0][21:19] =
3'ox;
    test_vecs[0][18:16] = 3'h3; test_vecs[0][15:0] = 16'hcdef;
    test_vecs[1][25] = 1'b1; test_vecs[1][24:22] = 3'ox; test_vecs[1][21:19] =
3'ox;
    test_vecs[1][18:16] = 3'o7; test_vecs[1][15:0] = 16'h3210;
    test_vecs[2][25] = 1'b1; test_vecs[2][24:22] = 3'o3; test_vecs[2][21:19] =
3'o7;
    test_vecs[2][18:16] = 3'o5; test_vecs[2][15:0] = 16'h4567;
    test_vecs[3][25] = 1'b1; test_vecs[3][24:22] = 3'o1; test_vecs[3][21:19] =
3'o5;
    test_vecs[3][18:16] = 3'o0; test_vecs[3][15:0] = 16'hba98;
    test_vecs[4][25] = 1'b0; test_vecs[4][24:22] = 3'o1; test_vecs[4][21:19] =
3'o5;
    test_vecs[4][18:16] = 3'o1; test_vecs[4][15:0] = 16'hxxxx;
    test_vecs[5][25] = 1'b0; test_vecs[5][24:22] = 3'o0; test_vecs[5][21:19] =
3'o0;
    test_vecs[5][18:16] = 3'ox; test_vecs[5][15:0] = 16'hxxxx;
  end
  initial {wr, rd_addr_a, rd_addr_b, wr_addr, d_in} = 0;
  reg_file reg_file_0 (clk, reset, wr, rd_addr_a, rd_addr_b, wr_addr, d_in,
d_out_a, d_out_b);
  initial begin
```

```
    #6 for(i=0;i<`TESTVECS;i=i+1)
      begin #10 {wr, rd_addr_a, rd_addr_b, wr_addr, d_in}=test_vecs[i]; end
    #100 $finish;
  end
endmodule
```

Output: