Name: Adithya M SRN:PES1UG20CS621 SEC: K

reg_alu.v

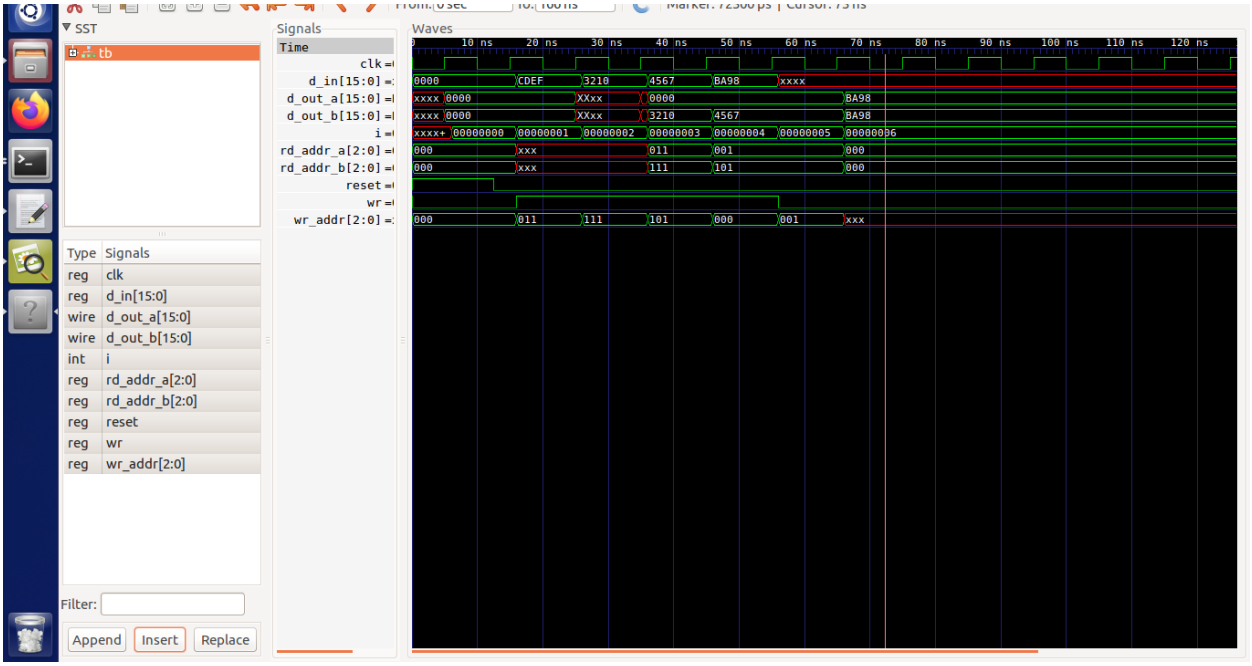
```
module fa (input wire i0, i1, cin, output wire sum, cout);
   wire t0, t1, t2;
   xor3 _i0 (i0, i1, cin, sum);
   and2 _i1 (i0, i1, t0);
   and2 _i2 (i1, cin, t1);
   and2 _i3 (cin, i0, t2);
   or3 _i4 (t0, t1, t2, cout);
endmodule

module addsub (input wire addsub, i0, i1, cin, output wire sumdiff, cout);
  wire t;
  fa _i0 (i0, t, cin, sumdiff, cout);
  xor2 _i1 (i1, addsub, t);
endmodule

module alu_slice (input wire [1:0] op, input wire i0, i1, cin, output wire o, cout);
   wire t_sumdiff, t_and, t_or, t_andor;
   addsub _i0 (op[0], i0, i1, cin, t_sumdiff, cout);
   and2 _i1 (i0, i1, t_and);
   or2 _i2 (i0, i1, t_or);
   mux2 _i3 (t_and, t_or, op[0], t_andor);
   mux2 _i4 (t_sumdiff, t_andor, op[1], o);
endmodule

module alu (input wire [1:0] op, input wire [15:0] i0, i1,
    output wire [15:0] o, output wire cout);
   wire [14:0] c;
   alu_slice _i0 (op, i0[0], i1[0], op[0] , o[0], c[0]);
   alu_slice _i1 (op, i0[1], i1[1], c[0], o[1], c[1]);
   alu_slice _i2 (op, i0[2], i1[2], c[1], o[2], c[2]);
   alu_slice _i3 (op, i0[3], i1[3], c[2], o[3], c[3]);
   alu_slice _i4 (op, i0[4], i1[4], c[3], o[4], c[4]);
   alu_slice _i5 (op, i0[5], i1[5], c[4], o[5], c[5]);
   alu_slice _i6 (op, i0[6], i1[6], c[5], o[6], c[6]);
   alu_slice _i7 (op, i0[7], i1[7], c[6], o[7], c[7]);
   alu_slice _i8 (op, i0[8], i1[8], c[7], o[8], c[8]);
   alu_slice _i9 (op, i0[9], i1[9], c[8], o[9], c[9]);
   alu_slice _i10 (op, i0[10], i1[10], c[9] , o[10], c[10]);
   alu_slice _i11 (op, i0[11], i1[11], c[10], o[11], c[11]);
   alu_slice _i12 (op, i0[12], i1[12], c[11], o[12], c[12]);
   alu_slice _i13 (op, i0[13], i1[13], c[12], o[13], c[13]);
   alu_slice _i14 (op, i0[14], i1[14], c[13], o[14], c[14]);
   alu_slice _i15 (op, i0[15], i1[15], c[14], o[15], cout);
endmodule


// Write code for modules you need here
module reg16(input wire clk,reset,load,input wire [15:0]din,output wire [15:0]r);
        dfrl d0(clk,reset,load,din[0],r[0]);
        dfrl d1(clk,reset,load,din[1],r[1]);
        dfrl d2(clk,reset,load,din[2],r[2]);
        dfrl d3(clk,reset,load,din[3],r[3]);
        dfrl d4(clk,reset,load,din[4],r[4]);
        dfrl d5(clk,reset,load,din[5],r[5]);
```

```
        dfrl d6(clk,reset,load,din[6],r[6]);
        dfrl d7(clk,reset,load,din[7],r[7]);
        dfrl d8(clk,reset,load,din[8],r[8]);
        dfrl d9(clk,reset,load,din[9],r[9]);
        dfrl d10(clk,reset,load,din[10],r[10]);
        dfrl d11(clk,reset,load,din[11],r[11]);
        dfrl d12(clk,reset,load,din[12],r[12]);
        dfrl d13(clk,reset,load,din[13],r[13]);
        dfrl d14(clk,reset,load,din[14],r[14]);
        dfrl d15(clk,reset,load,din[15],r[15]);
endmodule

module mux128_16(input wire [15:0]i0,i1,i2,i3,i4,i5,i6,i7,input wire s0,s1,s2,output wire [15:0]o);

        mux8 mx0({i0[0],i1[0],i2[0],i3[0],i4[0],i5[0],i6[0],i7[0]},s2,s1,s0,o[0]);
        mux8 mx1({i0[1],i1[1],i2[1],i3[1],i4[1],i5[1],i6[1],i7[1]},s2,s1,s0,o[1]);
        mux8 mx2({i0[2],i1[2],i2[2],i3[2],i4[2],i5[2],i6[2],i7[2]},s2,s1,s0,o[2]);
        mux8 mx3({i0[3],i1[3],i2[3],i3[3],i4[3],i5[3],i6[3],i7[3]},s2,s1,s0,o[3]);
        mux8 mx4({i0[4],i1[4],i2[4],i3[4],i4[4],i5[4],i6[4],i7[4]},s2,s1,s0,o[4]);
        mux8 mx5({i0[5],i1[5],i2[5],i3[5],i4[5],i5[5],i6[5],i7[5]},s2,s1,s0,o[5]);
        mux8 mx6({i0[6],i1[6],i2[6],i3[6],i4[6],i5[6],i6[6],i7[6]},s2,s1,s0,o[6]);
        mux8 mx7({i0[7],i1[7],i2[7],i3[7],i4[7],i5[7],i6[7],i7[7]},s2,s1,s0,o[7]);
        mux8 mx8({i0[8],i1[8],i2[8],i3[8],i4[8],i5[8],i6[8],i7[8]},s2,s1,s0,o[8]);
        mux8 mx9({i0[9],i1[9],i2[9],i3[9],i4[9],i5[9],i6[9],i7[9]},s2,s1,s0,o[9]);
        mux8 mx10({i0[10],i1[10],i2[10],i3[10],i4[10],i5[10],i6[10],i7[10]},s2,s1,s0,o[10]);
        mux8 mx11({i0[11],i1[11],i2[11],i3[11],i4[11],i5[11],i6[11],i7[11]},s2,s1,s0,o[11]);
        mux8 mx12({i0[12],i1[12],i2[12],i3[12],i4[12],i5[12],i6[12],i7[12]},s2,s1,s0,o[12]);
        mux8 mx13({i0[13],i1[13],i2[13],i3[13],i4[13],i5[13],i6[13],i7[13]},s2,s1,s0,o[13]);
        mux8 mx14({i0[14],i1[14],i2[14],i3[14],i4[14],i5[14],i6[14],i7[14]},s2,s1,s0,o[14]);
        mux8 mx15({i0[15],i1[15],i2[15],i3[15],i4[15],i5[15],i6[15],i7[15]},s2,s1,s0,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);

        mux128_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);
        mux128_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);


module mux2for16(input wire [15:0] din_regular, alu_out, input wire selector, output wire [15:0]din_final);
        mux2 m0(din_regular[0], alu_out[0], selector, din_final[0]);
        mux2 m1(din_regular[1], alu_out[1], selector, din_final[1]);
        mux2 m2(din_regular[2], alu_out[2], selector, din_final[2]);
        mux2 m3(din_regular[3], alu_out[3], selector, din_final[3]);
        mux2 m4(din_regular[4], alu_out[4], selector, din_final[4]);
        mux2 m5(din_regular[5], alu_out[5], selector, din_final[5]);
        mux2 m6(din_regular[6], alu_out[6], selector, din_final[6]);
        mux2 m7(din_regular[7], alu_out[7], selector, din_final[7]);
        mux2 m8(din_regular[8], alu_out[8], selector, din_final[8]);
        mux2 m9(din_regular[9], alu_out[9], selector, din_final[9]);
        mux2 m10(din_regular[10], alu_out[10], selector, din_final[10]);
        mux2 m11(din_regular[11], alu_out[11], selector, din_final[11]);
        mux2 m12(din_regular[12], alu_out[12], selector, din_final[12]);
        mux2 m13(din_regular[13], alu_out[13], selector, din_final[13]);
        mux2 m14(din_regular[14], alu_out[14], selector, din_final[14]);
        mux2 m15(din_regular[15], alu_out[15], selector, din_final[15]);
endmodule


module reg_alu (input wire clk, reset, sel, wr, input wire [1:0] op, 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, output wire cout);
        wire [15:0] alu_out;
        wire [15:0] newdin;
        mux2for16 select(d_in, alu_out, sel, newdin);
        reg_file new_reg(clk, reset, wr, rd_addr_a, rd_addr_b, wr_addr, newdin, d_out_a, d_out_b);
        alu calc(op, d_out_a, d_out_b, alu_out, cout);

endmodule
```

tb_alu.v

```
`timescale 1 ns / 100 ps
`define TESTVECS 8

module tb;
  reg clk, reset, wr, sel;
  reg [1:0] op;
  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 [28:0] test_vecs [0:(`TESTVECS-1)];
  integer i;
  initial begin $dumpfile("tb_reg_alu.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][28] = 1'b0; test_vecs[0][27] = 1'b1; test_vecs[0][26:25] = 2'bxx;
    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][28] = 1'b0; test_vecs[1][27] = 1'b1; test_vecs[1][26:25] = 2'bxx;
    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][28] = 1'b0; test_vecs[2][27] = 1'b1; test_vecs[2][26:25] = 2'bxx;
    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][28] = 1'b0; test_vecs[3][27] = 1'b1; test_vecs[3][26:25] = 2'bxx;
    test_vecs[3][24:22] = 3'o1; test_vecs[3][21:19] = 3'o5;
    test_vecs[3][18:16] = 3'o1; test_vecs[3][15:0] = 16'hba98;

    test_vecs[4][28] = 1'b0; test_vecs[4][27] = 1'b0; test_vecs[4][26:25] = 2'bxx;
    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][28] = 1'b1; test_vecs[5][27] = 1'b1; test_vecs[5][26:25] = 2'b00;
    test_vecs[5][24:22] = 3'o1; test_vecs[5][21:19] = 3'o5;
    test_vecs[5][18:16] = 3'o2; test_vecs[5][15:0] = 16'hxxxx;

    test_vecs[6][28] = 1'b1; test_vecs[6][27] = 1'b1; test_vecs[6][26:25] = 2'b01;
    test_vecs[6][24:22] = 3'o3; test_vecs[6][21:19] = 3'o7;
    test_vecs[6][18:16] = 3'o4; test_vecs[6][15:0] = 16'hxxxx;

    test_vecs[7][28] = 1'b1; test_vecs[7][27] = 1'b0; test_vecs[7][26:25] = 2'b01;
    test_vecs[7][24:22] = 3'o0; test_vecs[7][21:19] = 3'o0;
    test_vecs[7][18:16] = 3'ox; test_vecs[7][15:0] = 16'hxxxx;
  end
  initial {sel, wr, op, rd_addr_a, rd_addr_b, wr_addr, d_in} = 0;
  reg_alu reg_alu_0 (clk, reset, sel, wr, op, rd_addr_a, rd_addr_b, wr_addr, d_in,
  d_out_a, d_out_b, cout);
  initial begin
    #6 for(i=0;i<`TESTVECS;i=i+1)
      begin #10 {sel, wr, op, rd_addr_a, rd_addr_b, wr_addr, d_in}=test_vecs[i]; end
    #100 $finish;
  end
endmodule
```

Output:


SST
tb
Type Signals
reg clk
reg d_in[15:0]
wire d_out_a[15:0]
wire d_out_b[15:0]
int i
reg rd_addr_a[2:0]
reg rd_addr_b[2:0]
reg reset
reg wr
reg wr_addr[2:0]
Filter:
Append
Insert
Replace
Signals
Time
clk
d_in[15:0]
d_out_a[15:0]
d_out_b[15:0]
i
rd_addr_a[2:0]
rd_addr_b[2:0]
reset
wr
wr_addr[2:0]
Waves
10 ns
20 ns
30 ns
40 ns
50 ns
60 ns
70 ns
80 ns
90 ns
100 ns
110 ns
120 ns