**Digital Design and Computer Organization Laboratory**

**UE19CS206**

**3rd Semester, Academic Year 2020-21**

Date: 11-10-2021

|  |  |  |
| --- | --- | --- |
| Name : M.H.SOHAN | SRN : PES1UG20CS235 | Section : 3 D |

Experiment Number: 5 Week # : 5

**Title of the Program: Integration Of ALU with the Register File**

**Code:**

**Reg\_Alu.v:**

module dfrl\_reg (input wire clk,reset,load,input wire[15:0]i,output wire [15:0]out);

dfrl f0(clk,reset,load,i[0],out[0]);

dfrl f1(clk,reset,load,i[1],out[1]);

dfrl f2(clk,reset,load,i[2],out[2]);

dfrl f3(clk,reset,load,i[3],out[3]);

dfrl f4(clk,reset,load,i[4],out[4]);

dfrl f5(clk,reset,load,i[5],out[5]);

dfrl f6(clk,reset,load,i[6],out[6]);

dfrl f7(clk,reset,load,i[7],out[7]);

dfrl f8(clk,reset,load,i[8],out[8]);

dfrl f9(clk,reset,load,i[9],out[9]);

dfrl f10(clk,reset,load,i[10],out[10]);

dfrl f11(clk,reset,load,i[11],out[11]);

dfrl f12(clk,reset,load,i[12],out[12]);

dfrl f13(clk,reset,load,i[13],out[13]);

dfrl f14(clk,reset,load,i[14],out[14]);

dfrl f15(clk,reset,load,i[15],out[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 m\_0({i0[0], i1[0], i2[0], i3[0], i4[0], i5[0], i6[0], i7[0]},j0, j1, j2, o[0]);

mux8 m\_1({i0[1], i1[1], i2[1], i3[1], i4[1], i5[1], i6[1], i7[1]},j0, j1, j2, o[1]);

mux8 m\_2({i0[2], i1[2], i2[2], i3[2], i4[2], i5[2], i6[2], i7[2]},j0, j1, j2, o[2]);

mux8 m\_3({i0[3], i1[3], i2[3], i3[3], i4[3], i5[3], i6[3], i7[3]},j0, j1, j2, o[3]);

mux8 m\_4({i0[4], i1[4], i2[4], i3[4], i4[4], i5[4], i6[4], i7[4]},j0, j1, j2, o[4]);

mux8 m\_5({i0[5], i1[5], i2[5], i3[5], i4[5], i5[5], i6[5], i7[5]},j0, j1, j2, o[5]);

mux8 m\_6({i0[6], i1[6], i2[6], i3[6], i4[6], i5[6], i6[6], i7[6]},j0, j1, j2, o[6]);

mux8 m\_7({i0[7], i1[7], i2[7], i3[7], i4[7], i5[7], i6[7], i7[7]},j0, j1, j2, o[7]);

mux8 m\_8({i0[8], i1[8], i2[8], i3[8], i4[8], i5[8], i6[8], i7[8]},j0, j1, j2, o[8]);

mux8 m\_9({i0[9], i1[9], i2[9], i3[9], i4[9], i5[9], i6[9], i7[9]},j0, j1, j2, o[9]);

mux8 m\_10({i0[10], i1[10], i2[10], i3[10], i4[10], i5[10], i6[10], i7[10]},j0, j1, j2, o[10]);

mux8 m\_11({i0[11], i1[11], i2[11], i3[11], i4[11], i5[11], i6[11], i7[11]},j0, j1, j2, o[11]);

mux8 m\_12({i0[12], i1[12], i2[12], i3[12], i4[12], i5[12], i6[12], i7[12]},j0, j1, j2, o[12]);

mux8 m\_13({i0[13], i1[13], i2[13], i3[13], i4[13], i5[13], i6[13], i7[13]},j0, j1, j2, o[13]);

mux8 m\_14({i0[14], i1[14], i2[14], i3[14], i4[14], i5[14], i6[14], i7[14]},j0, j1, j2, o[14]);

mux8 m\_15({i0[15], i1[15], i2[15], i3[15], i4[15], i5[15], i6[15], i7[15]},j0, j1, j2, 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

dfrl\_reg reg\_0(clk,reset,load[0],d\_in,r0);

dfrl\_reg reg\_1(clk,reset,load[1],d\_in,r1);

dfrl\_reg reg\_2(clk,reset,load[2],d\_in,r2);

dfrl\_reg reg\_3(clk,reset,load[3],d\_in,r3);

dfrl\_reg reg\_4(clk,reset,load[4],d\_in,r4);

dfrl\_reg reg\_5(clk,reset,load[5],d\_in,r5);

dfrl\_reg reg\_6(clk,reset,load[6],d\_in,r6);

dfrl\_reg reg\_7(clk,reset,load[7],d\_in,r7);

demux8 dem(wr,wr\_addr[2],wr\_addr[1],wr\_addr[0],load);

mux8\_16 m0(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 m1(r0,r1,r2,r3,r4,r5,r6,r7,rd\_addr\_a[0],rd\_addr\_a[1],rd\_addr\_a[2],d\_out\_b);

endmodule

module mux2to1\_16(input wire [15:0] d\_in, o, input wire sel, output wire [15:0]o1);

mux2 mux2f0(d\_in[0],o[0],sel,o1[0]);

mux2 mux2f1(d\_in[1],o[1],sel,o1[1]);

mux2 mux2f2(d\_in[2],o[2],sel,o1[2]);

mux2 mux2f3(d\_in[3],o[3],sel,o1[3]);

mux2 mux2f4(d\_in[4],o[4],sel,o1[4]);

mux2 mux2f5(d\_in[5],o[5],sel,o1[5]);

mux2 mux2f6(d\_in[6],o[6],sel,o1[6]);

mux2 mux2f7(d\_in[7],o[7],sel,o1[7]);

mux2 mux2f8(d\_in[8],o[8],sel,o1[8]);

mux2 mux2f9(d\_in[9],o[9],sel,o1[9]);

mux2 mux2f10(d\_in[10],o[10],sel,o1[10]);

mux2 mux2f11(d\_in[11],o[11],sel,o1[11]);

mux2 mux2f12(d\_in[12],o[12],sel,o1[12]);

mux2 mux2f13(d\_in[13],o[13],sel,o1[13]);

mux2 mux2f14(d\_in[14],o[14],sel,o1[14]);

mux2 mux2f15(d\_in[15],o[15],sel,o1[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);

// Declare wires here

wire [15:0] o;

wire [15:0] o1;

wire [15:0] o2;

wire [15:0] o3;

// Instantiate modules here

reg\_file regf0(clk,reset,wr,rd\_addr\_a, rd\_addr\_b, wr\_addr,d\_in,o2, o3);

alu aluf0(op,o2, o3,o,cout);

mux2to1\_16 select1\_16(d\_in,o,sel,o1);

reg\_file regf1(clk,reset,wr,rd\_addr\_a, rd\_addr\_b, wr\_addr,d\_in,d\_out\_a, d\_out\_b);

endmodule

**alu.v:**

module fa (input wire i0, i1, cin, output wire sum, cout);

wire t0;

wire t1;

wire 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;

wire t\_and;

wire t\_or;

wire 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

**Output waveform:**

Graphical user interface

Description automatically generated with low confidence