Digital Design and Computer Organization Laboratory

UE23CS251A

3rd Semester, Academic Year 2024-25

Date:29/10/2024

Name: Keerthan P.V	SRN: PES2UG23CS272	Section:E

Week : 7 Program Number : 1

Title of the Program

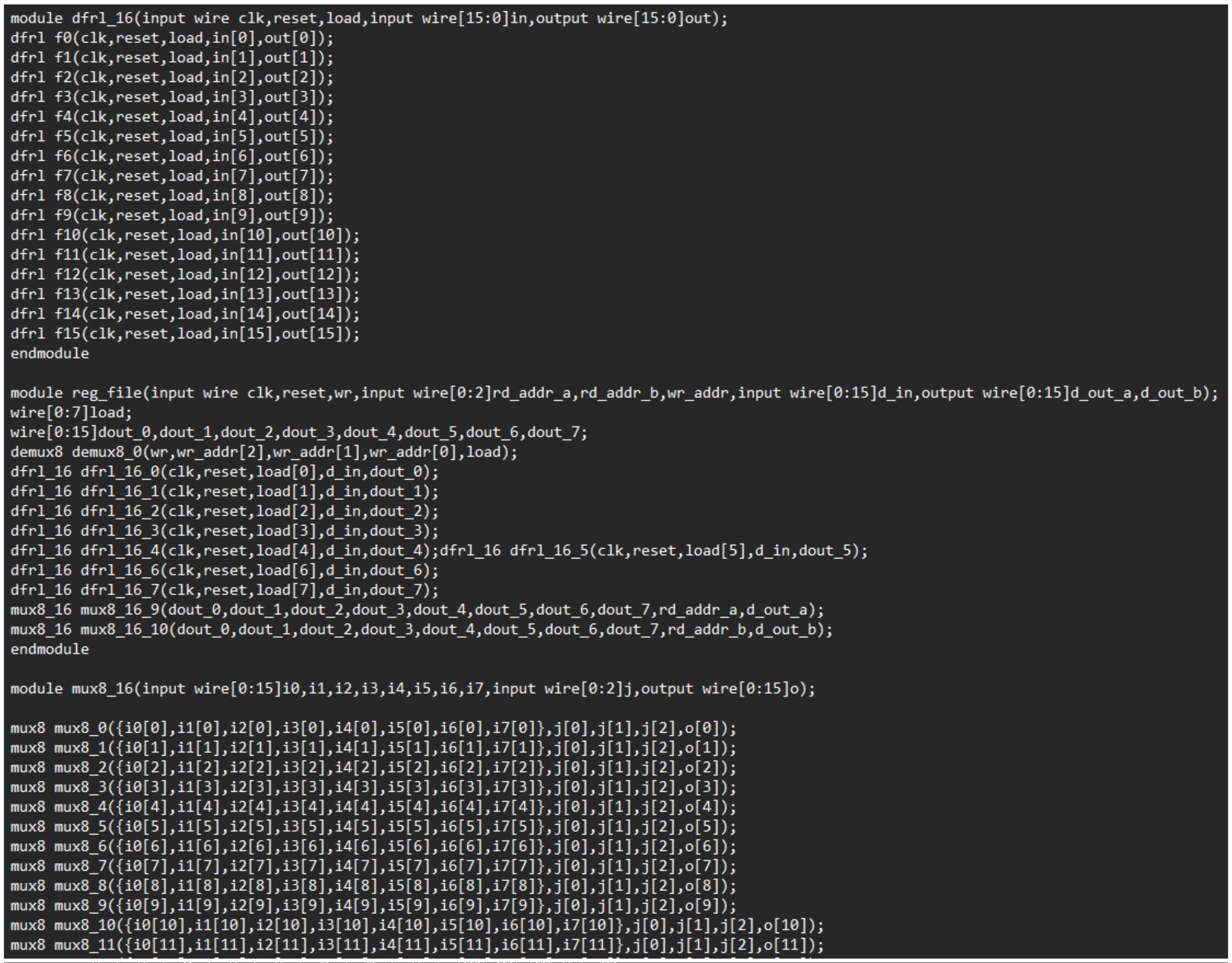
<u>INTEGRATION OF ALU AND REG_FILE TO FORM REG_ALU.</u>

AIM: TO CONSTRUCT A REGISTER FILE, FROM WHICH TWO 16-BIT VALUES CAN BE READ, AND TO WHICH ONE 16-BIT VALUE WRITTEN, EVERY CLOCK CYCLE. THE REGISTER FILE THUS NEEDS TO BE INTEGRATED WITH THE ALU

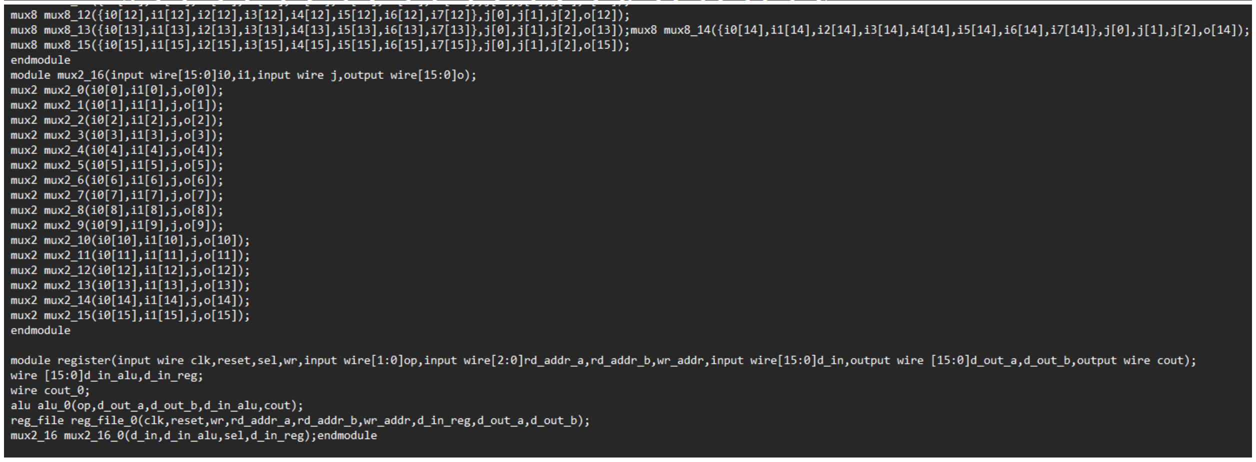
THE TWO READ OUTPUTS AND THE WRITE INPUT OF THE OF THE REGISTER FILE IMPLEMENTED NEED TO BE CONNECTED RESPECTIVELY TO THE TWO INPUTS AND ONE OUTPUT OF THE ALU

1: Paste the Screen Shot of the source code, test bench in reg_alu.v

Main file code:

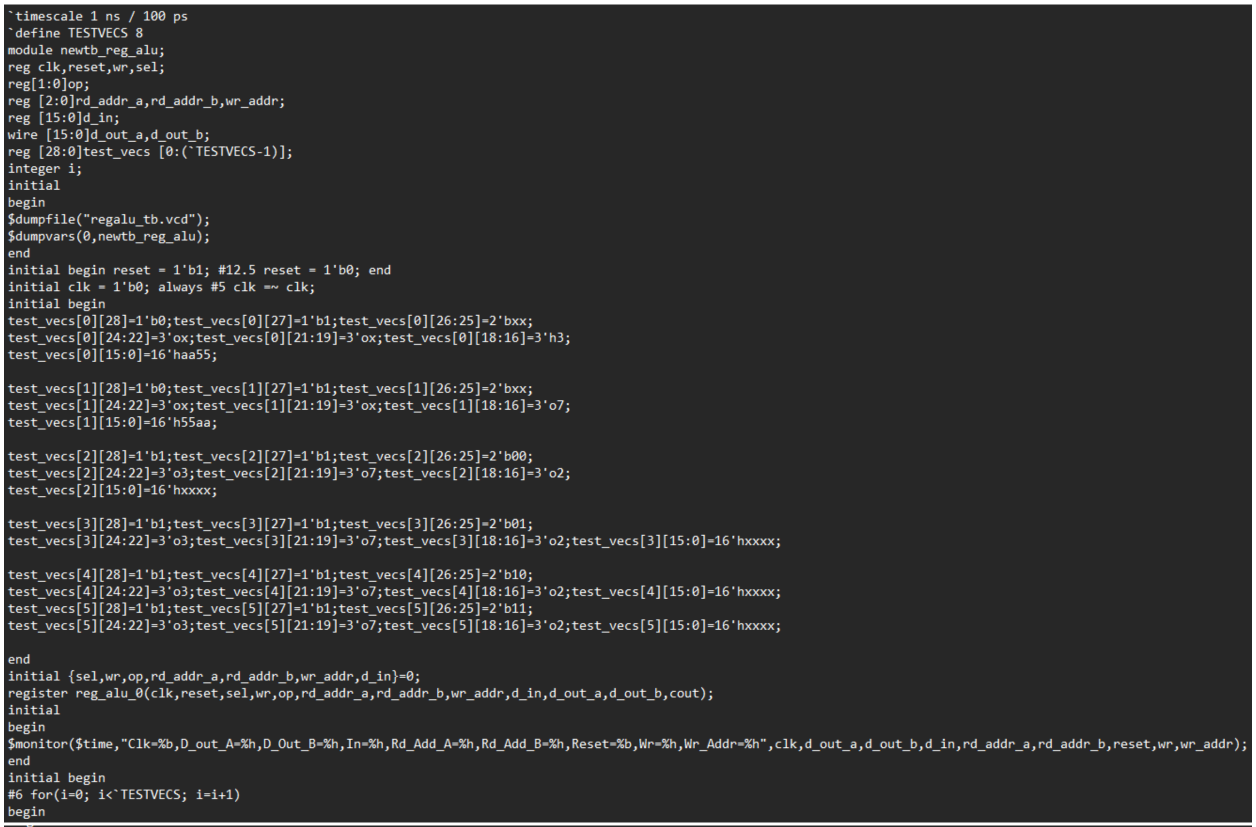


```
module dfrl_16(input wire clk,reset,load,input wire[15:0]in,output wire[15:0]out);
dfrl f0(clk,reset,load,in[0],out[0]);
dfrl f1(clk,reset,load,in[1],out[1]);
dfrl f2(clk,reset,load,in[2],out[2]);
dfrl f3(clk,reset,load,in[3],out[3]);
dfrl f4(clk,reset,load,in[4],out[4]);
dfrl f5(clk,reset,load,in[5],out[5]);
dfrl f6(clk,reset,load,in[6],out[6]);
dfrl f7(clk,reset,load,in[7],out[7]);
dfrl f8(clk,reset,load,in[8],out[8]);
dfrl f9(clk,reset,load,in[9],out[9]);
dfrl f10(clk,reset,load,in[10],out[10]);
dfrl f11(clk,reset,load,in[11],out[11]);
dfrl f12(clk,reset,load,in[12],out[12]);
dfrl f13(clk,reset,load,in[13],out[13]);
dfrl f14(clk,reset,load,in[14],out[14]);
dfrl f15(clk,reset,load,in[15],out[15]);
endmodule

module reg_file(input wire clk,reset,wr,input wire[0:2]rd_addr_a,rd_addr_b,wr_addr,input wire[0:15]d_in,output wire[0:15]d_out_a,d_out_b);
wire[0:7]load;
wire[0:15]dout_0,dout_1,dout_2,dout_3,dout_4,dout_5,dout_6,dout_7;
demux8 demux8_0(wr,wr_addr[2],wr_addr[1],wr_addr[0],load);
dfrl_16 dfrl_16_0(clk,reset,load[0],d_in,dout_0);
dfrl_16 dfrl_16_1(clk,reset,load[1],d_in,dout_1);
dfrl_16 dfrl_16_2(clk,reset,load[2],d_in,dout_2);
dfrl_16 dfrl_16_3(clk,reset,load[3],d_in,dout_3);
dfrl_16 dfrl_16_4(clk,reset,load[4],d_in,dout_4);dfrl_16 dfrl_16_5(clk,reset,load[5],d_in,dout_5);
dfrl_16 dfrl_16_6(clk,reset,load[6],d_in,dout_6);
dfrl_16 dfrl_16_7(clk,reset,load[7],d_in,dout_7);
mux8_16 mux8_16_9(dout_0,dout_1,dout_2,dout_3,dout_4,dout_5,dout_6,dout_7,rd_addr_a,d_out_a);
mux8_16 mux8_16_10(dout_0,dout_1,dout_2,dout_3,dout_4,dout_5,dout_6,dout_7,rd_addr_b,d_out_b);
endmodule

module mux8_16(input wire[0:15]i0,i1,i2,i3,i4,i5,i6,i7,input wire[0:2]j,output wire[0:15]o);

mux8 mux8_0({i0[0],i1[0],i2[0],i3[0],i4[0],i5[0],i6[0],i7[0]},j[0],j[1],j[2],o[0]);
mux8 mux8_1({i0[1],i1[1],i2[1],i3[1],i4[1],i5[1],i6[1],i7[1]},j[0],j[1],j[2],o[1]);
mux8 mux8_2({i0[2],i1[2],i2[2],i3[2],i4[2],i5[2],i6[2],i7[2]},j[0],j[1],j[2],o[2]);
mux8 mux8_3({i0[3],i1[3],i2[3],i3[3],i4[3],i5[3],i6[3],i7[3]},j[0],j[1],j[2],o[3]);
mux8 mux8_4({i0[4],i1[4],i2[4],i3[4],i4[4],i5[4],i6[4],i7[4]},j[0],j[1],j[2],o[4]);
mux8 mux8_5({i0[5],i1[5],i2[5],i3[5],i4[5],i5[5],i6[5],i7[5]},j[0],j[1],j[2],o[5]);
mux8 mux8_6({i0[6],i1[6],i2[6],i3[6],i4[6],i5[6],i6[6],i7[6]},j[0],j[1],j[2],o[6]);
mux8 mux8_7({i0[7],i1[7],i2[7],i3[7],i4[7],i5[7],i6[7],i7[7]},j[0],j[1],j[2],o[7]);
mux8 mux8_8({i0[8],i1[8],i2[8],i3[8],i4[8],i5[8],i6[8],i7[8]},j[0],j[1],j[2],o[8]);
mux8 mux8_9({i0[9],i1[9],i2[9],i3[9],i4[9],i5[9],i6[9],i7[9]},j[0],j[1],j[2],o[9]);
mux8 mux8_10({i0[10],i1[10],i2[10],i3[10],i4[10],i5[10],i6[10],i7[10]},j[0],j[1],j[2],o[10]);
mux8 mux8_11({i0[11],i1[11],i2[11],i3[11],i4[11],i5[11],i6[11],i7[11]},j[0],j[1],j[2],o[11]);
```

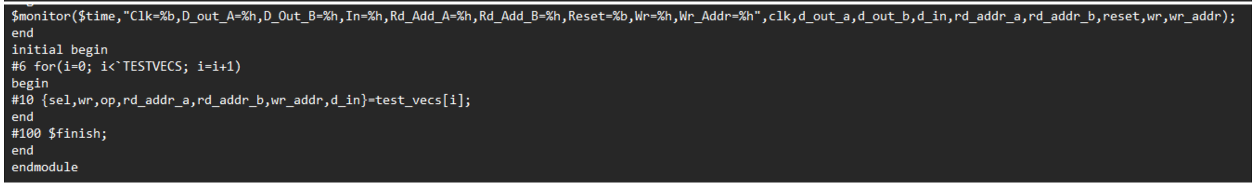


```
mux8 mux8_12({i0[12],i1[12],i2[12],i3[12],i4[12],i5[12],i6[12],i7[12]},j[0],j[1],j[2],o[12]);
mux8 mux8_13({i0[13],i1[13],i2[13],i3[13],i4[13],i5[13],i6[13],i7[13]},j[0],j[1],j[2],o[13]);mux8 mux8_14({i0[14],i1[14],i2[14],i3[14],i4[14],i5[14],i6[14],i7[14]},j[0],j[1],j[2],o[14]);
mux8 mux8_15({i0[15],i1[15],i2[15],i3[15],i4[15],i5[15],i6[15],i7[15]},j[0],j[1],j[2],o[15]);
endmodule
module mux2_16(input wire[15:0]i0,i1,input wire j,output wire[15:0]o);
mux2 mux2_0(i0[0],i1[0],j,o[0]);
mux2 mux2_1(i0[1],i1[1],j,o[1]);
mux2 mux2_2(i0[2],i1[2],j,o[2]);
mux2 mux2_3(i0[3],i1[3],j,o[3]);
mux2 mux2_4(i0[4],i1[4],j,o[4]);
mux2 mux2_5(i0[5],i1[5],j,o[5]);
mux2 mux2_6(i0[6],i1[6],j,o[6]);
mux2 mux2_7(i0[7],i1[7],j,o[7]);
mux2 mux2_8(i0[8],i1[8],j,o[8]);
mux2 mux2_9(i0[9],i1[9],j,o[9]);
mux2 mux2_10(i0[10],i1[10],j,o[10]);
mux2 mux2_11(i0[11],i1[11],j,o[11]);
mux2 mux2_12(i0[12],i1[12],j,o[12]);
mux2 mux2_13(i0[13],i1[13],j,o[13]);
mux2 mux2_14(i0[14],i1[14],j,o[14]);
mux2 mux2_15(i0[15],i1[15],j,o[15]);
endmodule

module register(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]d_in_alu,d_in_reg;
wire cout_0;
alu alu_0(op,d_out_a,d_out_b,d_in_alu,cout);
reg_file reg_file_0(clk,reset,wr,rd_addr_a,rd_addr_b,wr_addr,d_in_reg,d_out_a,d_out_b);
mux2_16 mux2_16_0(d_in,d_in_alu,sel,d_in_reg);endmodule
```

Testbench file code:



```
`timescale 1 ns / 100 ps
`define TESTVECS 8
module newtb_reg_alu;
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("regalu_tb.vcd");
$dumpvars(0,newtb_reg_alu);
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'haa55;

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'h55aa;

test_vecs[2][28]=1'b1;test_vecs[2][27]=1'b1;test_vecs[2][26:25]=2'b00;
test_vecs[2][24:22]=3'o3;test_vecs[2][21:19]=3'o7;test_vecs[2][18:16]=3'o2;
test_vecs[2][15:0]=16'hxxxx;

test_vecs[3][28]=1'b1;test_vecs[3][27]=1'b1;test_vecs[3][26:25]=2'b01;
test_vecs[3][24:22]=3'o3;test_vecs[3][21:19]=3'o7;test_vecs[3][18:16]=3'o2;test_vecs[3][15:0]=16'hxxxx;

test_vecs[4][28]=1'b1;test_vecs[4][27]=1'b1;test_vecs[4][26:25]=2'b10;
test_vecs[4][24:22]=3'o3;test_vecs[4][21:19]=3'o7;test_vecs[4][18:16]=3'o2;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'b11;
test_vecs[5][24:22]=3'o3;test_vecs[5][21:19]=3'o7;test_vecs[5][18:16]=3'o2;test_vecs[5][15:0]=16'hxxxx;

end
initial {sel,wr,op,rd_addr_a,rd_addr_b,wr_addr,d_in}=0;
register 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
$monitor($time,"Clk=%b,D_out_A=%h,D_Out_B=%h,In=%h,Rd_Add_A=%h,Rd_Add_B=%h,Reset=%b,Wr=%h,Wr_Addr=%h",clk,d_out_a,d_out_b,d_in,rd_addr_a,rd_addr_b,reset,wr,wr_addr);
end
initial begin
#6 for(i=0; i<`TESTVECS; i=i+1)
begin
```



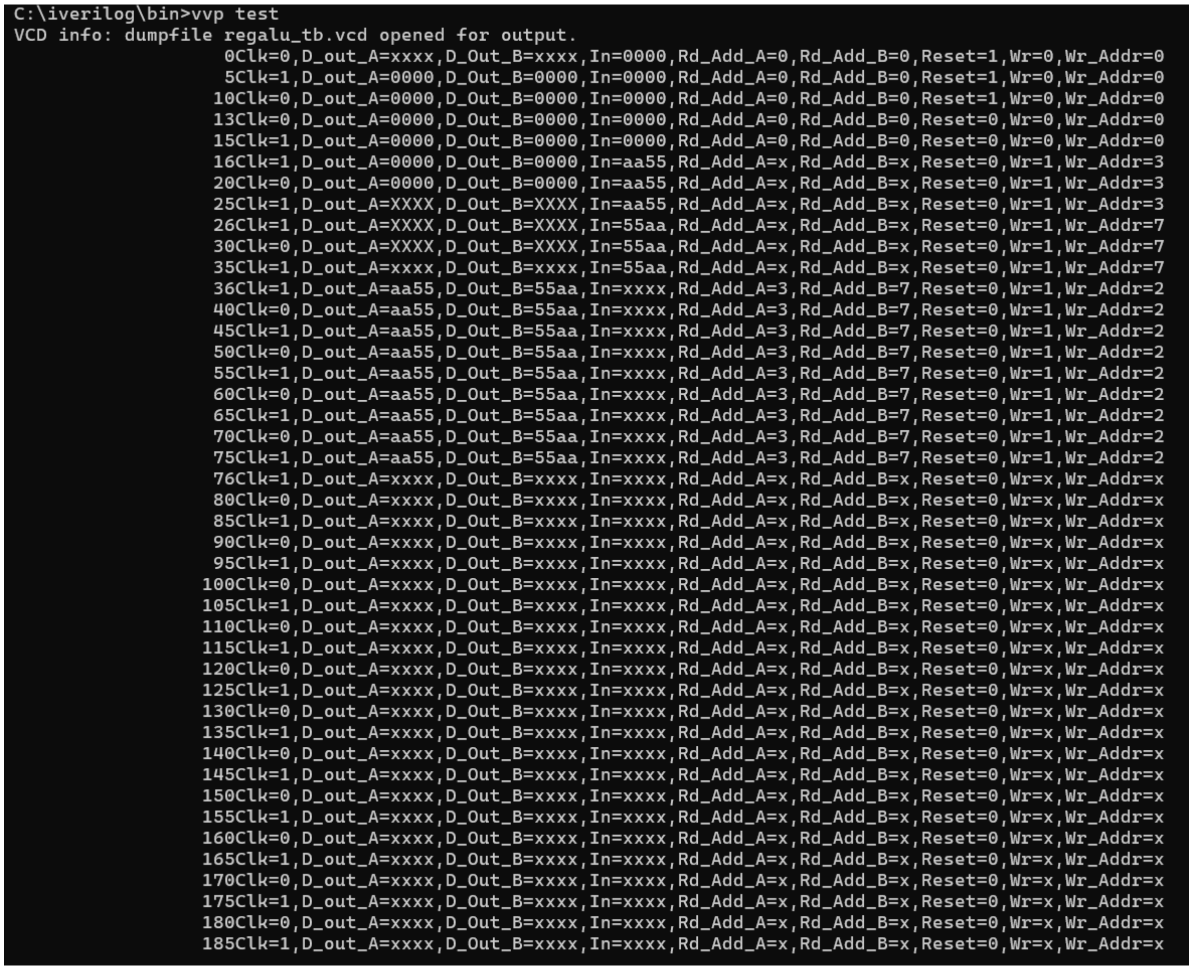
```
$monitor($time,"Clk=%b,D_out_A=%h,D_Out_B=%h,In=%h,Rd_Add_A=%h,Rd_Add_B=%h,Reset=%b,Wr=%h,Wr_Addr=%h",clk,d_out_a,d_out_b,d_in,rd_addr_a,rd_addr_b,reset,wr,wr_addr);
end
initial begin
#6 for(i=0; i<`TESTVECS; i=i+1)
begin
```

```
#10 {sel,wr,op,rd_addr_a,rd_addr_b,wr_addr,d_in}=test_vecs[i];
end
#100 $finish;
end
endmodule
```

2. Complete the truth table

sel	wr	op	rd_addr_a			rd_addr_b			wr_addr			d_in	Output
28	27	26-25	24	23	22	21	20	19	18	17	16	Bit15 to Bit 0	
0	1	xx	x	x	x	x	x	x	0	1	1	AA55	Reg3=AA55
0	1	xx	x	x	x	x	x	x	1	1	1	55AA	Reg7=55AA
1	1	00	0	1	1	1	1	1	0	1	0	XXXX	Reg2=FFFF
1	1	01	0	1	1	1	1	1	0	1	0	XXXX	Reg2=54AB
1	1	10	0	1	1	1	1	1	0	1	0	XXXX	Reg2=0000
1	1	11	0	1	1	1	1	1	0	1	0	XXXX	Reg2=FFFF

3.VVP Output



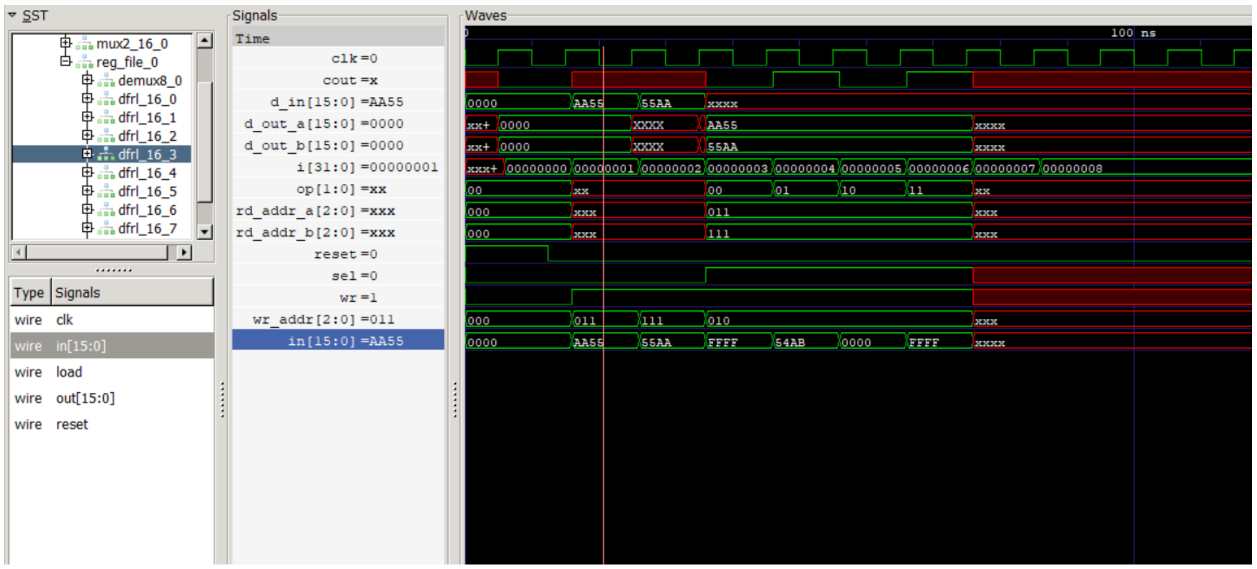
```
C:\iverilog\bin>vvp test
VCD info: dumpfile regalu_tb.vcd opened for output.
                   0Clk=0,D_out_A=xxxx,D_Out_B=xxxx,In=0000,Rd_Add_A=0,Rd_Add_B=0,Reset=1,Wr=0,Wr_Addr=0
                   5Clk=1,D_out_A=0000,D_Out_B=0000,In=0000,Rd_Add_A=0,Rd_Add_B=0,Reset=1,Wr=0,Wr_Addr=0
                  10Clk=0,D_out_A=0000,D_Out_B=0000,In=0000,Rd_Add_A=0,Rd_Add_B=0,Reset=1,Wr=0,Wr_Addr=0
                  13Clk=0,D_out_A=0000,D_Out_B=0000,In=0000,Rd_Add_A=0,Rd_Add_B=0,Reset=0,Wr=0,Wr_Addr=0
                  15Clk=1,D_out_A=0000,D_Out_B=0000,In=0000,Rd_Add_A=0,Rd_Add_B=0,Reset=0,Wr=0,Wr_Addr=0
                  16Clk=1,D_out_A=0000,D_Out_B=0000,In=aa55,Rd_Add_A=x,Rd_Add_B=x,Reset=0,Wr=1,Wr_Addr=3
                  20Clk=0,D_out_A=0000,D_Out_B=0000,In=aa55,Rd_Add_A=x,Rd_Add_B=x,Reset=0,Wr=1,Wr_Addr=3
                  25Clk=1,D_out_A=XXXX,D_Out_B=XXXX,In=aa55,Rd_Add_A=x,Rd_Add_B=x,Reset=0,Wr=1,Wr_Addr=3
                  26Clk=1,D_out_A=XXXX,D_Out_B=XXXX,In=55aa,Rd_Add_A=x,Rd_Add_B=x,Reset=0,Wr=1,Wr_Addr=7
                  30Clk=0,D_out_A=XXXX,D_Out_B=XXXX,In=55aa,Rd_Add_A=x,Rd_Add_B=x,Reset=0,Wr=1,Wr_Addr=7
                  35Clk=1,D_out_A=xxxx,D_Out_B=xxxx,In=55aa,Rd_Add_A=x,Rd_Add_B=x,Reset=0,Wr=1,Wr_Addr=7
                  36Clk=1,D_out_A=aa55,D_Out_B=55aa,In=xxxx,Rd_Add_A=3,Rd_Add_B=7,Reset=0,Wr=1,Wr_Addr=2
                  40Clk=0,D_out_A=aa55,D_Out_B=55aa,In=xxxx,Rd_Add_A=3,Rd_Add_B=7,Reset=0,Wr=1,Wr_Addr=2
                  45Clk=1,D_out_A=aa55,D_Out_B=55aa,In=xxxx,Rd_Add_A=3,Rd_Add_B=7,Reset=0,Wr=1,Wr_Addr=2
                  50Clk=0,D_out_A=aa55,D_Out_B=55aa,In=xxxx,Rd_Add_A=3,Rd_Add_B=7,Reset=0,Wr=1,Wr_Addr=2
                  55Clk=1,D_out_A=aa55,D_Out_B=55aa,In=xxxx,Rd_Add_A=3,Rd_Add_B=7,Reset=0,Wr=1,Wr_Addr=2
                  60Clk=0,D_out_A=aa55,D_Out_B=55aa,In=xxxx,Rd_Add_A=3,Rd_Add_B=7,Reset=0,Wr=1,Wr_Addr=2
                  65Clk=1,D_out_A=aa55,D_Out_B=55aa,In=xxxx,Rd_Add_A=3,Rd_Add_B=7,Reset=0,Wr=1,Wr_Addr=2
                  70Clk=0,D_out_A=aa55,D_Out_B=55aa,In=xxxx,Rd_Add_A=3,Rd_Add_B=7,Reset=0,Wr=1,Wr_Addr=2
                  75Clk=1,D_out_A=aa55,D_Out_B=55aa,In=xxxx,Rd_Add_A=3,Rd_Add_B=7,Reset=0,Wr=1,Wr_Addr=2
                  76Clk=1,D_out_A=xxxx,D_Out_B=xxxx,In=xxxx,Rd_Add_A=x,Rd_Add_B=x,Reset=0,Wr=x,Wr_Addr=x
                  80Clk=0,D_out_A=xxxx,D_Out_B=xxxx,In=xxxx,Rd_Add_A=x,Rd_Add_B=x,Reset=0,Wr=x,Wr_Addr=x
                  85Clk=1,D_out_A=xxxx,D_Out_B=xxxx,In=xxxx,Rd_Add_A=x,Rd_Add_B=x,Reset=0,Wr=x,Wr_Addr=x
                  90Clk=0,D_out_A=xxxx,D_Out_B=xxxx,In=xxxx,Rd_Add_A=x,Rd_Add_B=x,Reset=0,Wr=x,Wr_Addr=x
                  95Clk=1,D_out_A=xxxx,D_Out_B=xxxx,In=xxxx,Rd_Add_A=x,Rd_Add_B=x,Reset=0,Wr=x,Wr_Addr=x
                 100Clk=0,D_out_A=xxxx,D_Out_B=xxxx,In=xxxx,Rd_Add_A=x,Rd_Add_B=x,Reset=0,Wr=x,Wr_Addr=x
                 105Clk=1,D_out_A=xxxx,D_Out_B=xxxx,In=xxxx,Rd_Add_A=x,Rd_Add_B=x,Reset=0,Wr=x,Wr_Addr=x
                 110Clk=0,D_out_A=xxxx,D_Out_B=xxxx,In=xxxx,Rd_Add_A=x,Rd_Add_B=x,Reset=0,Wr=x,Wr_Addr=x
                 115Clk=1,D_out_A=xxxx,D_Out_B=xxxx,In=xxxx,Rd_Add_A=x,Rd_Add_B=x,Reset=0,Wr=x,Wr_Addr=x
                 120Clk=0,D_out_A=xxxx,D_Out_B=xxxx,In=xxxx,Rd_Add_A=x,Rd_Add_B=x,Reset=0,Wr=x,Wr_Addr=x
                 125Clk=1,D_out_A=xxxx,D_Out_B=xxxx,In=xxxx,Rd_Add_A=x,Rd_Add_B=x,Reset=0,Wr=x,Wr_Addr=x
                 130Clk=0,D_out_A=xxxx,D_Out_B=xxxx,In=xxxx,Rd_Add_A=x,Rd_Add_B=x,Reset=0,Wr=x,Wr_Addr=x
                 135Clk=1,D_out_A=xxxx,D_Out_B=xxxx,In=xxxx,Rd_Add_A=x,Rd_Add_B=x,Reset=0,Wr=x,Wr_Addr=x
                 140Clk=0,D_out_A=xxxx,D_Out_B=xxxx,In=xxxx,Rd_Add_A=x,Rd_Add_B=x,Reset=0,Wr=x,Wr_Addr=x
                 145Clk=1,D_out_A=xxxx,D_Out_B=xxxx,In=xxxx,Rd_Add_A=x,Rd_Add_B=x,Reset=0,Wr=x,Wr_Addr=x
                 150Clk=0,D_out_A=xxxx,D_Out_B=xxxx,In=xxxx,Rd_Add_A=x,Rd_Add_B=x,Reset=0,Wr=x,Wr_Addr=x
                 155Clk=1,D_out_A=xxxx,D_Out_B=xxxx,In=xxxx,Rd_Add_A=x,Rd_Add_B=x,Reset=0,Wr=x,Wr_Addr=x
                 160Clk=0,D_out_A=xxxx,D_Out_B=xxxx,In=xxxx,Rd_Add_A=x,Rd_Add_B=x,Reset=0,Wr=x,Wr_Addr=x
                 165Clk=1,D_out_A=xxxx,D_Out_B=xxxx,In=xxxx,Rd_Add_A=x,Rd_Add_B=x,Reset=0,Wr=x,Wr_Addr=x
                 170Clk=0,D_out_A=xxxx,D_Out_B=xxxx,In=xxxx,Rd_Add_A=x,Rd_Add_B=x,Reset=0,Wr=x,Wr_Addr=x
                 175Clk=1,D_out_A=xxxx,D_Out_B=xxxx,In=xxxx,Rd_Add_A=x,Rd_Add_B=x,Reset=0,Wr=x,Wr_Addr=x
                 180Clk=0,D_out_A=xxxx,D_Out_B=xxxx,In=xxxx,Rd_Add_A=x,Rd_Add_B=x,Reset=0,Wr=x,Wr_Addr=x
                 185Clk=1,D_out_A=xxxx,D_Out_B=xxxx,In=xxxx,Rd_Add_A=x,Rd_Add_B=x,Reset=0,Wr=x,Wr_Addr=x
```

4.Paste the Screen shot of the GTKWave form

I. SCREENSHOT 1 :

CASE 1 (Write operation): sel =0,wr=1,Write Address=011,d_in=AA55,

Verify in[15:0] of Register 3



```
SST
  mux2_16_0
  reg_file_0
    demux8_0
    dfrl_16_0
    dfrl_16_1
    dfrl_16_2
    dfrl_16_3
    dfrl_16_4
    dfrl_16_5
    dfrl_16_6
    dfrl_16_7

Type  Signals
wire  clk
wire  in[15:0]
wire  load
wire  out[15:0]
wire  reset
```

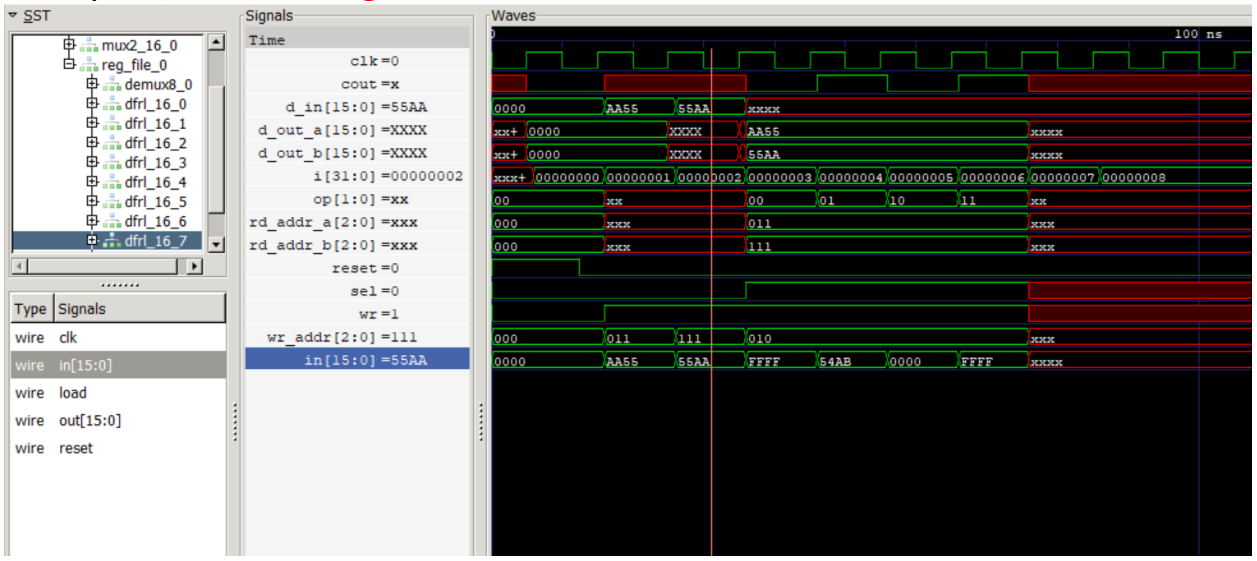
```
Signals
Time
           clk =0
          cout =x
     d_in[15:0] =AA55
  d_out_a[15:0] =0000
  d_out_b[15:0] =0000
        i[31:0] =00000001
        op[1:0] =xx
  rd_addr_a[2:0] =xxx
  rd_addr_b[2:0] =xxx
          reset =0
            sel =0
             wr =1
   wr_addr[2:0] =011
      in[15:0] =AA55
```

II. SCREENSHOT 2 :

CASE 2 (Write operation):

sel =0,wr=1 ,Write Address=111,d_in=55AA,

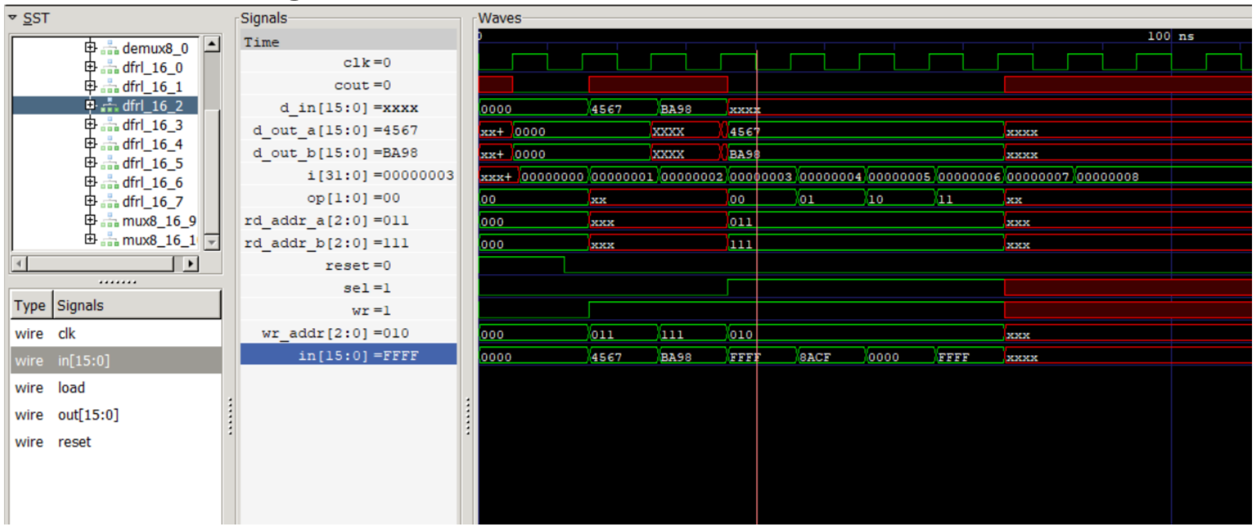
Verify in[15:0] of Register 7



```
SST
  mux2_16_0
  reg_file_0
    demux8_0
    dfrl_16_0
    dfrl_16_1
    dfrl_16_2
    dfrl_16_3
    dfrl_16_4
    dfrl_16_5
    dfrl_16_6
    dfrl_16_7

Type  Signals
wire  clk
wire  in[15:0]
wire  load
wire  out[15:0]
wire  reset
```

```
Signals
Time
           clk =0
          cout =x
     d_in[15:0] =55AA
  d_out_a[15:0] =XXXX
  d_out_b[15:0] =XXXX
        i[31:0] =00000002
        op[1:0] =xx
  rd_addr_a[2:0] =xxx
  rd_addr_b[2:0] =xxx
          reset =0
            sel =0
             wr =1
   wr_addr[2:0] =111
      in[15:0] =55AA
```

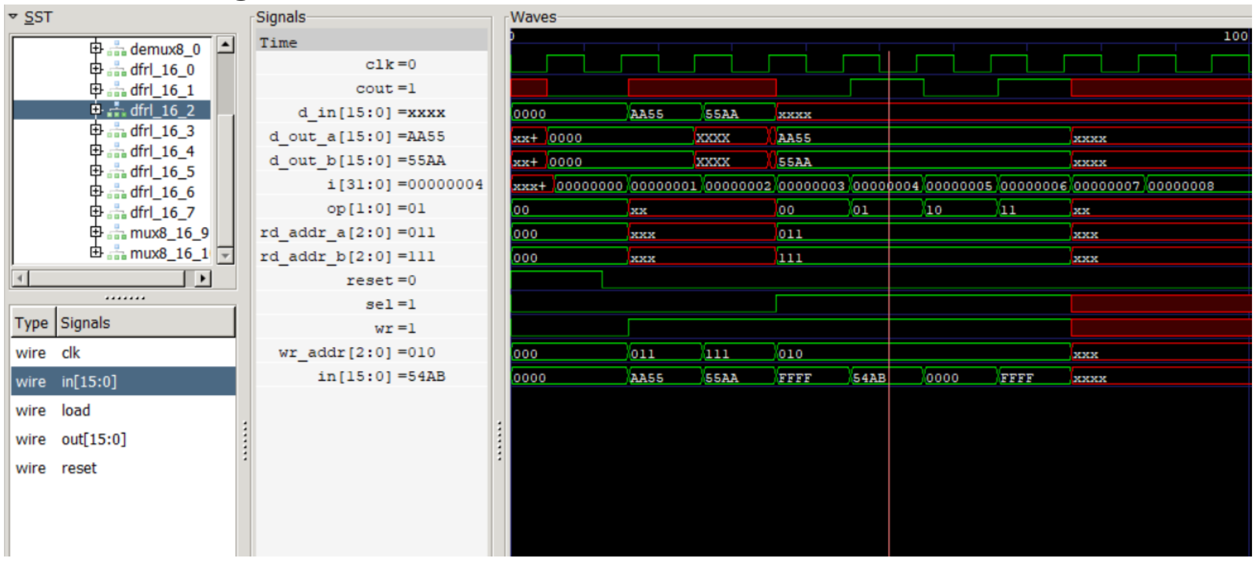
III. SCREENSHOT 3:
CASE 3 (Write operation after addition):
sel =1,wr=1,op=00, rd_addr_a=011, rd_addr_b=111,wr_addr= 010 d_in = XXXX,ALU output=d_out_a + d_out_b=4567+BA98=FFFF to be written to Reg2



```
SST
demux8_0
dfrl_16_0
dfrl_16_1
dfrl_16_2
dfrl_16_3
dfrl_16_4
dfrl_16_5
dfrl_16_6
dfrl_16_7
mux8_16_9
mux8_16_1

Type Signals
wire clk
wire in[15:0]
wire load
wire out[15:0]
wire reset
```

```
Signals
Time
clk =0
cout =0
d_in[15:0] =xxxx
d_out_a[15:0] =4567
d_out_b[15:0] =BA98
i[31:0] =00000003
op[1:0] =00
rd_addr_a[2:0] =011
rd_addr_b[2:0] =111
reset =0
sel =1
wr =1
wr_addr[2:0] =010
in[15:0] =FFFF
```

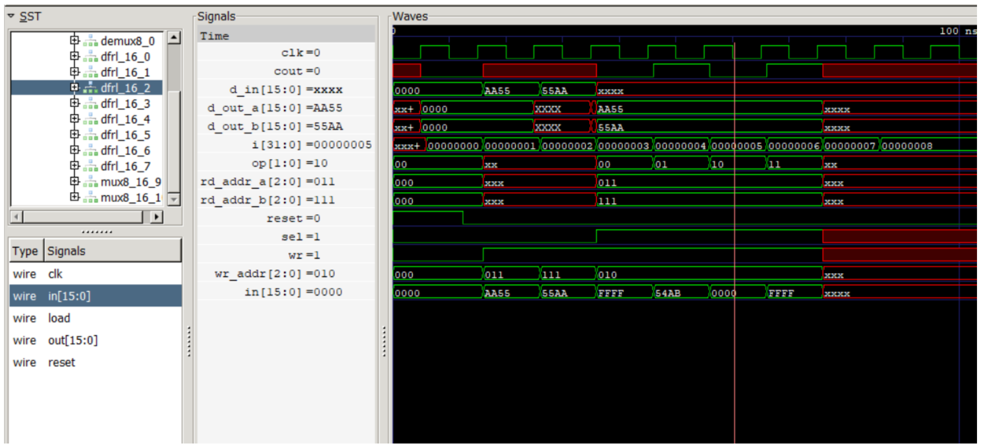
IV. SCREENSHOT 4 **CASE 4(Write operation after subtraction):**
sel =1,wr=1,op=01, rd_addr_a=010, rd_addr_b=111,wr_addr= 010 d_in = XXXX,ALU output=Reg3-Reg7=AA55-55AA=54AB to be written to Reg2



```
SST
demux8_0
dfrl_16_0
dfrl_16_1
dfrl_16_2
dfrl_16_3
dfrl_16_4
dfrl_16_5
dfrl_16_6
dfrl_16_7
mux8_16_9
mux8_16_1

Type Signals
wire clk
wire in[15:0]
wire load
wire out[15:0]
wire reset
```

```
Signals
Time
clk =0
cout =1
d_in[15:0] =xxxx
d_out_a[15:0] =AA55
d_out_b[15:0] =55AA
i[31:0] =00000004
op[1:0] =01
rd_addr_a[2:0] =011
rd_addr_b[2:0] =111
reset =0
sel =1
wr =1
wr_addr[2:0] =010
in[15:0] =54AB
```

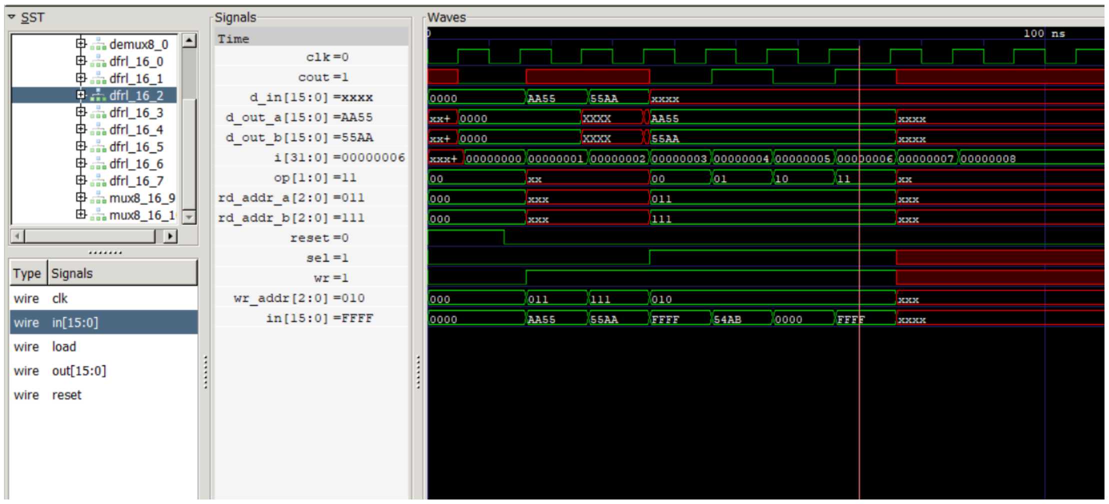
V. SCREENSHOT 5 :
CASE 5(Write operation after AND operation):
sel =1,wr=1,op=10, rd_addr_a=011, rd_addr_b=111,wr_addr= 010 d_in = XXXX,ALU output=Reg3 and Reg 7=AA55 And 55AA=0000 to be written to Reg2



Type	Signals
wire	clk
wire	in[15:0]
wire	load
wire	out[15:0]
wire	reset

```
Time
clk=0
cout=0
d_in[15:0]=xxxx
d_out_a[15:0]=AA55
d_out_b[15:0]=55AA
i[31:0]=00000005
op[1:0]=10
rd_addr_a[2:0]=011
rd_addr_b[2:0]=111
reset=0
sel=1
wr=1
wr_addr[2:0]=010
in[15:0]=0000
```

VI. SCREENSHOT 6 **CASE 6(Write operation after AND operation):**
sel =1,wr=1,op=11, rd_addr_a=011, rd_addr_b=111,wr_addr= 010 d_in = XXXX,ALU output=Reg3 OR Reg 7=AA55 OR 55AA=FFFF to be written to Reg2



Type	Signals
wire	clk
wire	in[15:0]
wire	load
wire	out[15:0]
wire	reset

```
Time
clk=0
cout=1
d_in[15:0]=xxxx
d_out_a[15:0]=AA55
d_out_b[15:0]=55AA
i[31:0]=00000006
op[1:0]=11
rd_addr_a[2:0]=011
rd_addr_b[2:0]=111
reset=0
sel=1
wr=1
wr_addr[2:0]=010
in[15:0]=FFFF
```

Disclaimer:

- The programs and output submitted is duly written, verified and executed my me.
- I have not copied from any of my peers nor from the external resource such as internet.
- If found plagiarized, I will abide with the disciplinary action of the University.

Signature: Keerthan P.V
Name: Keerthan P.V
SRN: PES2UG23CS272
Section: E
Date: 29/10/2024