Digital Design and Computer Organization Laboratory

UE23CS251A

3rd Semester, Academic Year 2024-25

Date: 29/10/2024

Name: ABHISHEK P	SRN:PES2UG23AM002	Section: A

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

Source code

```
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
);
   wire [15:0] r0, r1, r2, r3, r4, r5, r6, r7;
   reg16 reg0 (clk, reset, (wr && wr_addr == 3'b000), d_in, r0);
   reg16 reg1 (clk, reset, (wr && wr_addr == 3'b001), d_in, r1);
   reg16 reg2 (clk, reset, (wr && wr_addr == 3'b010), d_in, r2);
   reg16 reg3 (clk, reset, (wr && wr_addr == 3'b011), d_in, r3);
   reg16 reg4 (clk, reset, (wr && wr_addr == 3'b100), d_in, r4);
   reg16 reg5 (clk, reset, (wr && wr_addr == 3'b101), d_in, r5);
   reg16 reg6 (clk, reset, (wr && wr_addr == 3'b110), d_in, r6);
   reg16 reg7 (clk, reset, (wr && wr_addr == 3'b111), d_in, r7);

   mux8_16 mux_a (r0, r1, r2, r3, r4, r5, r6, r7, rd_addr_a, d_out_a);
   mux8_16 mux_b (r0, r1, r2, r3, r4, r5, r6, r7, rd_addr_b, d_out_b);
endmodule

module alu (
   input wire [15:0] a, b,
   input wire [1:0] op,
   output wire [15:0] result,
   output wire cout
);
   wire t_sumdiff, t_and, t_or, t_andor;
   wire [15:0] sumdiff_res, and_res, or_res;
   assign {cout, sumdiff_res} = a + b;
   assign and_res = a & b;
   assign or_res = a | b;
   assign t_sumdiff = (op[0] == 1'b0) ? sumdiff_res : and_res;
   assign t_andor = (op[0] == 1'b0) ? and_res : or_res;
   assign result = (op[1] == 1'b0) ? t_sumdiff : t_andor;
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] new_din;
   mux2for16 select (d_in, alu_out, sel, new_din);
   reg_file new_reg (
       .clk(clk), .reset(reset), .wr(wr),
       .rd_addr_a(rd_addr_a), .rd_addr_b(rd_addr_b),
       .wr_addr(wr_addr), .d_in(new_din),
       .d_out_a(d_out_a), .d_out_b(d_out_b)
   );
   alu calc (
       .a(d_out_a), .b(d_out_b), .op(op),
       .result(alu_out), .cout(cout)
   );
Endmodule
```

Testbench code

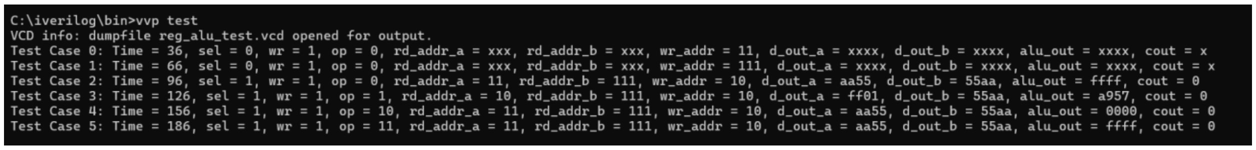
```
`timescale 1 ns / 100 ps
`define TESTVECS 6

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;
    wire cout;
    reg [28:0] test_vecs [0:(`TESTVECS-1)];
    integer i;

    initial begin
        $dumpfile("reg_alu_test.vcd");
        $dumpvars(0, tb);
    end
    initial begin
        reset = 1'b1;
        #12.5 reset = 1'b0;
    end

    initial clk = 1'b0;
    always #5 clk = ~clk;

    // Corrected test vectors for the cases described
    initial begin
        test_vecs[0] = {1'b0, 1'b1, 2'b00, 3'bxxx, 3'bxxx, 3'b011, 16'haa55}; // CASE1: Write AA55 to Register 3
        test_vecs[1] = {1'b0, 1'b1, 2'b00, 3'bxxx, 3'bxxx, 3'b111, 16'h55aa}; // CASE2: Write 55AA to Register 7
        test_vecs[2] = {1'b1, 1'b1, 2'b00, 3'b011, 3'b111, 3'b010, 16'hxxxx}; // CASE3: Add Reg3 (AA55) and Reg7 (55AA) to get FFFF, store in Reg2
        test_vecs[3] = {1'b1, 1'b1, 2'b01, 3'b010, 3'b111, 3'b010, 16'hxxxx}; // CASE4: Subtract Reg7 (55AA) from Reg2 (FFFF) to get 54AB, store in Reg2
        test_vecs[4] = {1'b1, 1'b1, 2'b10, 3'b011, 3'b111, 3'b010, 16'hxxxx}; // CASE5: AND Reg3 (AA55) and Reg7 (55AA) to get 0000, store in Reg2
        test_vecs[5] = {1'b1, 1'b1, 2'b11, 3'b011, 3'b111, 3'b010, 16'hxxxx}; // CASE6: OR Reg3 (AA55) and Reg7 (55AA) to get FFFF, store in Reg2
    end

    initial {sel, wr, op, rd_addr_a, rd_addr_b, wr_addr, d_in} = 0;

    reg_alu reg_alu_0 (
        .clk(clk),
        .reset(reset),
        .sel(sel),
        .wr(wr),
        .op(op),
        .rd_addr_a(rd_addr_a),
        .rd_addr_b(rd_addr_b),
        .wr_addr(wr_addr),
        .d_in(d_in),
        .d_out_a(d_out_a),
        .d_out_b(d_out_b),
        .cout(cout)
    );

    initial begin
        #6;
        for (i = 0; i < `TESTVECS; i = i + 1) begin
            {sel, wr, op, rd_addr_a, rd_addr_b, wr_addr, d_in} = test_vecs[i];
            #30; // Ensure sufficient time for updates

            // Display the state after each test case
            $display(
                "Test Case %0d: Time = %0d, sel = %0b, wr = %0b, op = %0b, rd_addr_a = %0b,
rd_addr_b = %0b, wr_addr = %0b, d_out_a = %h, d_out_b = %h, alu_out = %h, cout = %b",
                i, $time, sel, wr, op, rd_addr_a, rd_addr_b, wr_addr, d_out_a, d_out_b,
reg_alu_0.alu_out, cout
            );
        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



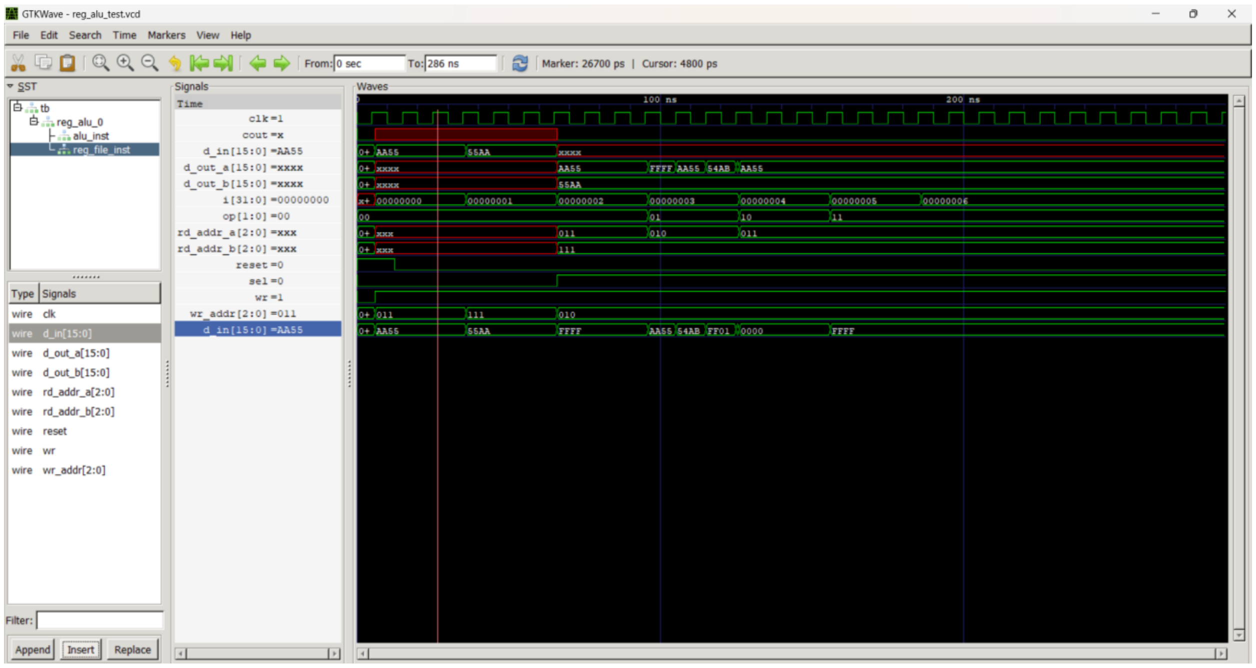
```
C:\iverilog\bin>vvp test
VCD info: dumpfile reg_alu_test.vcd opened for output.
Test Case 0: Time = 36, sel = 0, wr = 1, op = 0, rd_addr_a = xxx, rd_addr_b = xxx, wr_addr = 11, d_out_a = xxxx, d_out_b = xxxx, alu_out = xxxx, cout = x
Test Case 1: Time = 66, sel = 0, wr = 1, op = 0, rd_addr_a = xxx, rd_addr_b = xxx, wr_addr = 111, d_out_a = xxxx, d_out_b = xxxx, alu_out = xxxx, cout = x
Test Case 2: Time = 96, sel = 1, wr = 1, op = 0, rd_addr_a = 11, rd_addr_b = 111, wr_addr = 10, d_out_a = aa55, d_out_b = 55aa, alu_out = ffff, cout = 0
Test Case 3: Time = 126, sel = 1, wr = 1, op = 1, rd_addr_a = 10, rd_addr_b = 111, wr_addr = 10, d_out_a = ff01, d_out_b = 55aa, alu_out = a957, cout = 0
Test Case 4: Time = 156, sel = 1, wr = 1, op = 10, rd_addr_a = 11, rd_addr_b = 111, wr_addr = 10, d_out_a = aa55, d_out_b = 55aa, alu_out = 0000, cout = 0
Test Case 5: Time = 186, sel = 1, wr = 1, op = 11, rd_addr_a = 11, rd_addr_b = 111, wr_addr = 10, d_out_a = aa55, d_out_b = 55aa, alu_out = ffff, cout = 0
```

4.: Paste the Screen shot of the GTKWave form

I. SCREENSHOT1

CASE1 (Write operation):

sel =0,wr=1,Write Address=011,d_in=AA55,
Verify in[15:0] of Register 3


GTKWave - reg_alu_test.vcd
File Edit Search Time Markers View Help
SST
tb
reg_alu_0
alu_inst
reg_file_inst
Signals
Waves
100 ns
200 ns
Filter:
Append
Insert
Replace
0+ AA55 55AA xxxx
0+ xxxx AA55 FFFF AA55 54AB AA55
0+ xxxx 55AA
x+ 00000000 00000001 00000002 00000003 00000004 00000005 00000006
00 01 10 11
0+ xxx 011 010 011
0+ xxx 111
0+ 011 111 010
0+ AA55 55AA FFFF AA55 54AB FF01 0000 FFFF

From: 0 sec To: 286 ns Marker: 26700 ps | Cursor: 4800 ps

Type	Signals
wire	clk
wire	d_in[15:0]
wire	d_out_a[15:0]
wire	d_out_b[15:0]
wire	rd_addr_a[2:0]
wire	rd_addr_b[2:0]
wire	reset
wire	wr
wire	wr_addr[2:0]

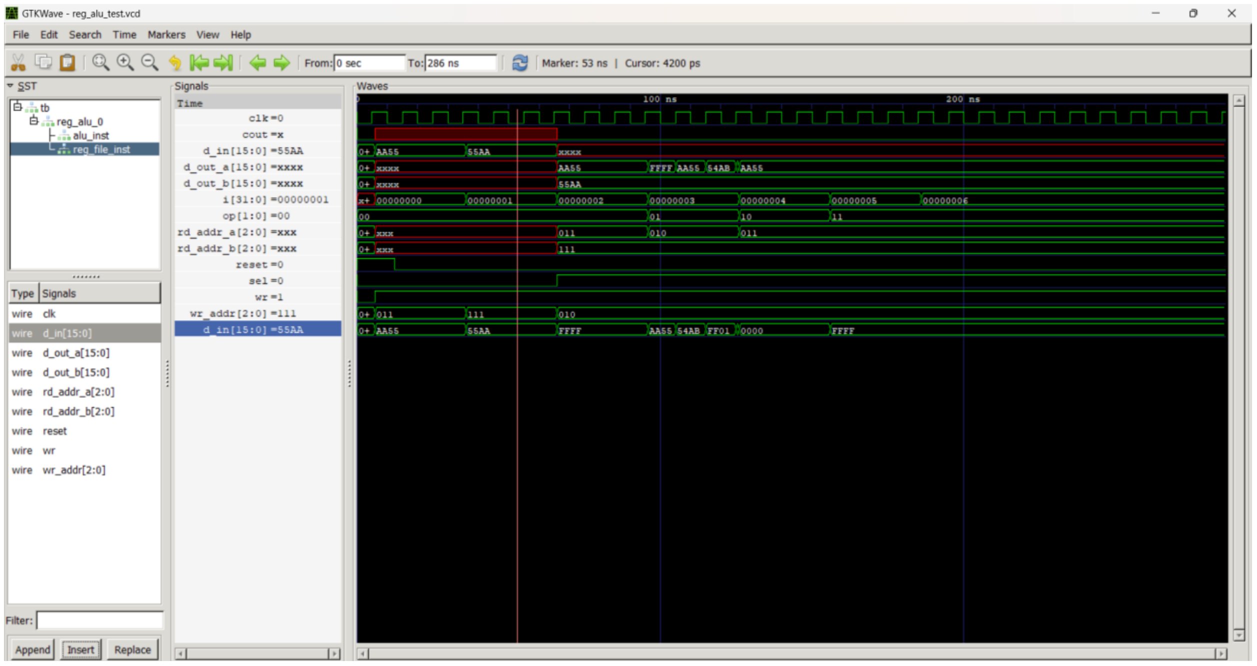
```
Time
clk=1
cout=x
d_in[15:0]=AA55
d_out_a[15:0]=xxxx
d_out_b[15:0]=xxxx
i[31:0]=00000000
op[1:0]=00
rd_addr_a[2:0]=xxx
rd_addr_b[2:0]=xxx
reset=0
sel=0
wr=1
wr_addr[2:0]=011
d_in[15:0]=AA55
```

II. SCREENSHOT 2

CASE2 (Write operation):

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

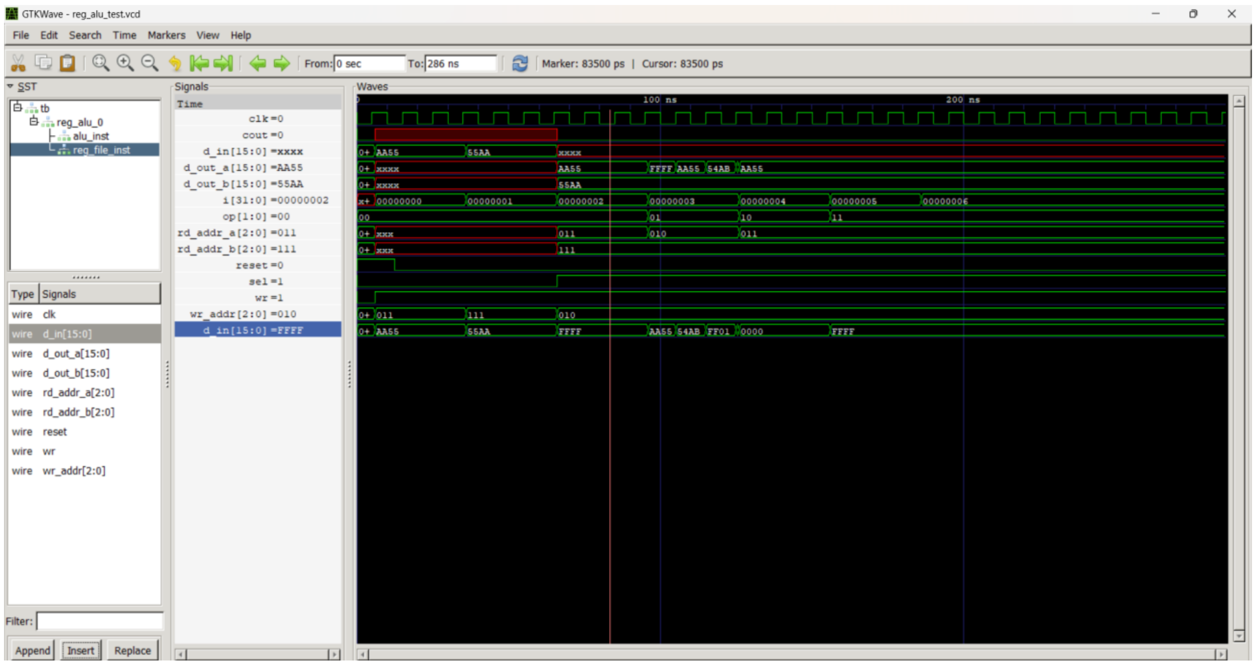
Verify in[15:0] of Register 7



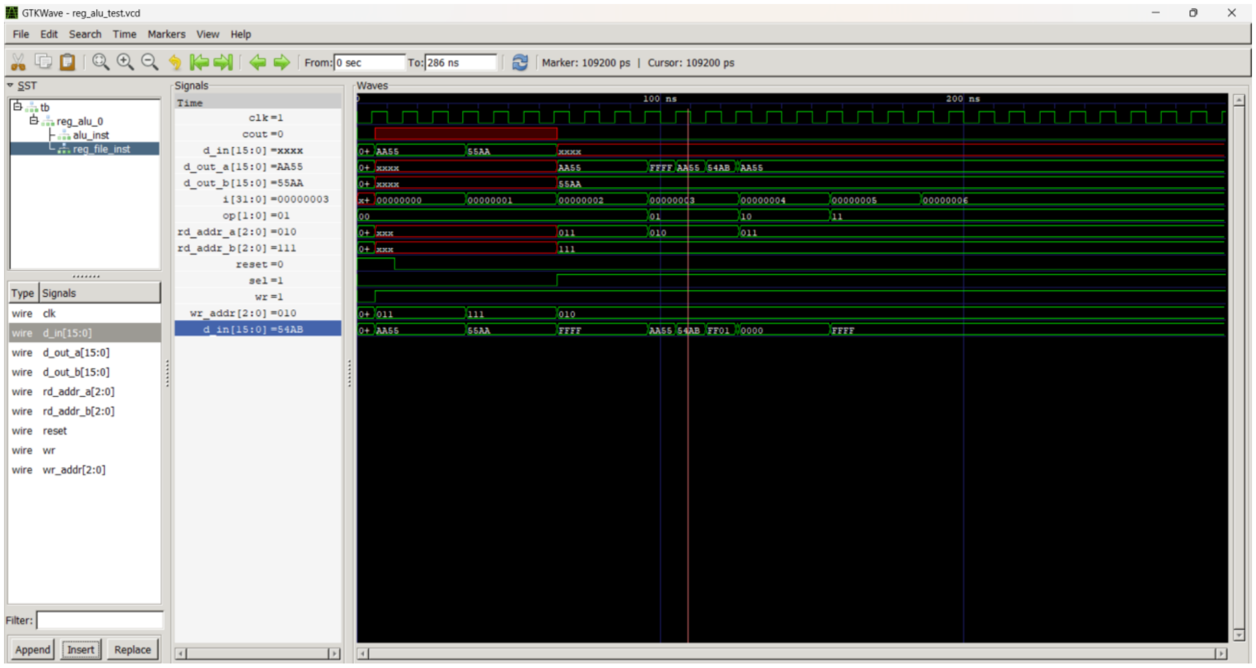
```
wire clk
wire d_in[15:0]
wire d_out_a[15:0]
wire d_out_b[15:0]
wire rd_addr_a[2:0]
wire rd_addr_b[2:0]
wire reset
wire wr
wire wr_addr[2:0]
```

```
Time
clk =0
cout =x
d_in[15:0] =55AA        AA55 | 55AA | xxxx
d_out_a[15:0] =xxxx     xxxx | AA55 | FFFF | AA55 | 54AB | AA55
d_out_b[15:0] =xxxx     xxxx | 55AA
i[31:0] =00000001       00000000 | 00000001 | 00000002 | 00000003 | 00000004 | 00000005 | 00000006
op[1:0] =00             00 | 01 | 10 | 11
rd_addr_a[2:0] =xxx     xxx | 011 | 010 | 011
rd_addr_b[2:0] =xxx     xxx | 111
reset =0
sel =0
wr =1
wr_addr[2:0] =111       011 | 111 | 010
d_in[15:0] =55AA        AA55 | 55AA | FFFF | AA55 | 54AB | FF01 | 0000 | FFFF
```

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

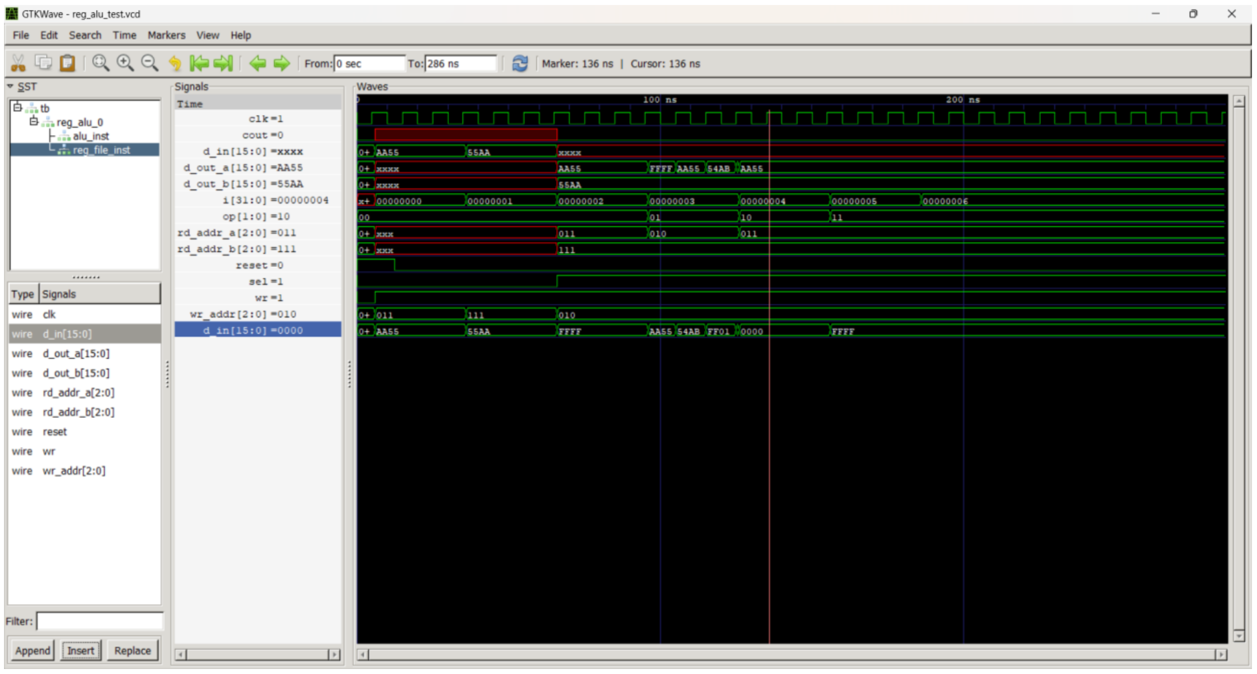


IV. SCREENSHOT 4 **CASE 4(Write operation after subtraction):**
sel =1,wr=1,op=01, rd_addr_a=010, rd_addr_b=111,wr_addr= 010 d_in = XXXX,ALU output=Reg3-Reg7=AA55-55AA=54AB to be written to Reg2



```
Time
clk =1
cout =0
d_in[15:0] =xxxx          0+ AA55 | 55AA | xxxx
d_out_a[15:0] =AA55       0+ xxxx | AA55 | FFFF | AA55 | 54AB | AA55
d_out_b[15:0] =55AA       0+ xxxx | 55AA
i[31:0] =00000003         x+ 00000000 | 00000001 | 00000002 | 00000003 | 00000004 | 00000005 | 00000006
op[1:0] =01               00 | 01 | 10 | 11
rd_addr_a[2:0] =010       0+ xxx | 011 | 010 | 011
rd_addr_b[2:0] =111       0+ xxx | 111
reset =0
sel =1
wr =1
wr_addr[2:0] =010         0+ 011 | 111 | 010
d_in[15:0] =54AB          0+ AA55 | 55AA | FFFF | AA55 | 54AB | FF01 | 0000 | FFFF
```

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



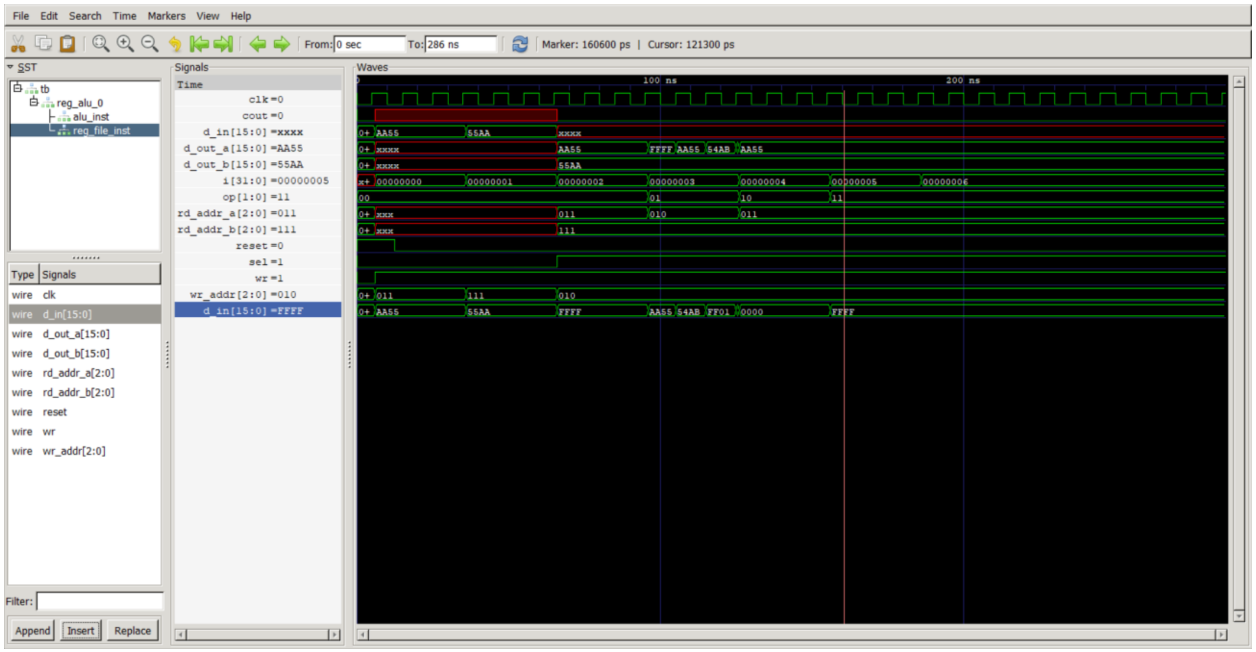
Marker: 136 ns | Cursor: 136 ns

Type	Signals
wire	clk
wire	d_in[15:0]
wire	d_out_a[15:0]
wire	d_out_b[15:0]
wire	rd_addr_a[2:0]
wire	rd_addr_b[2:0]
wire	reset
wire	wr
wire	wr_addr[2:0]

```
Time
clk =1
cout =0
d_in[15:0] =xxxx
d_out_a[15:0] =AA55
d_out_b[15:0] =55AA
i[31:0] =00000004
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
d_in[15:0] =0000
```

```
0+ AA55 55AA xxxx
0+ xxxx AA55 FFFF AA55 54AB AA55
0+ xxxx 55AA
x+ 00000000 00000001 00000002 00000003 00000004 00000005 00000006
00 01 10 11
0+ xxx 011 010 011
0+ xxx 111
0+ 011 111 010
0+ AA55 55AA FFFF AA55 54AB FF01 0000 FFFF
```

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



```
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] =11
rd_addr_a[2:0] =011
rd_addr_b[2:0] =111
reset =0
sel =1
wr =1
wr_addr[2:0] =010
d_in[15:0] =FFFF
```

```
wire clk
wire d_in[15:0]
wire d_out_a[15:0]
wire d_out_b[15:0]
wire rd_addr_a[2:0]
wire rd_addr_b[2:0]
wire reset
wire wr
wire wr_addr[2:0]
```

```
d_in[15:0]     0+ AA55 | 55AA | xxxx
d_out_a[15:0]  0+ xxxx | AA55 | FFFF | AA55 | 54AB | AA55
d_out_b[15:0]  0+ xxxx | 55AA
i[31:0]        x+ 00000000 | 00000001 | 00000002 | 00000003 | 00000004 | 00000005 | 00000006
op[1:0]        00 | 01 | 10 | 11
rd_addr_a[2:0] 0+ xxx | 011 | 010 | 011
rd_addr_b[2:0] 0+ xxx | 111
wr_addr[2:0]   0+ 011 | 111 | 010
d_in[15:0]     0+ AA55 | 55AA | FFFF | AA55 | 54AB | FF01 | 0000 | 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: [illegible]

Name: ABHISHEK P

SRN: PES2UG23AM002

Section: A

Date: 29/10/2024