Digital Design and Computer Organisation Laboratory

3rd Semester, Academic Year 2025

Date:13-10-2025

Name: VAIBHAV.R	SRN:PES2UG24CS664	Section:K

Week Number:8 Program Number: 1

TITLE : REGISTERS

Deliverables

I. Verilog Code Screenshot

Module 1

```
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 2

```
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 3

```
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 [7:0] load;
    wire [15:0] 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 d0(clk, reset, load[0], d_in, dout_0);
    dfrl_16 d1(clk, reset, load[1], d_in, dout_1);
    dfrl_16 d2(clk, reset, load[2], d_in, dout_2);
    dfrl_16 d3(clk, reset, load[3], d_in, dout_3);
    dfrl_16 d4(clk, reset, load[4], d_in, dout_4);
    dfrl_16 d5(clk, reset, load[5], d_in, dout_5);
    dfrl_16 d6(clk, reset, load[6], d_in, dout_6);
    dfrl_16 d7(clk, reset, load[7], d_in, dout_7);
    mux8_16 mux_a(dout_0, dout_1, dout_2, dout_3, dout_4, dout_5, dout_6, dout_7, rd_addr_a, d_out_a);
    mux8_16 mux_b(dout_0, dout_1, dout_2, dout_3, dout_4, dout_5, dout_6, dout_7, rd_addr_b, d_out_b);
endmodule
```

ALU

```
module reg_alu(
    input wire clk, reset, sel, wr,
    input wire [1:0] op,              // <-- match alu.v
    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] reg_a, reg_b, alu_out, reg_in;
    reg_file rf(
        .clk(clk),
        .reset(reset),
        .wr(wr),
        .rd_addr_a(rd_addr_a),
        .rd_addr_b(rd_addr_b),
        .wr_addr(wr_addr),
        .d_in(reg_in),
        .d_out_a(reg_a),
        .d_out_b(reg_b));
    alu alu0(
        .op(op),          // 2-bit
        .i0(reg_a),
        .i1(reg_b),
        .o(alu_out),
        .cout(cout));
    mux2_16 mux_sel(
        .sel(sel),
        .a(d_in),
        .b(alu_out),
        .y(reg_in));
    assign d_out_a = reg_a;
    assign d_out_b = reg_b;
endmodule
```

lib1.v

```
module invert (input wire i, output wire o);
  assign o = !i;
endmodule
module and2 (input wire i0, i1, output wire o);
 assign o = i0 & i1;
endmodule
module or2 (input wire i0, i1, output wire o);
 assign o = i0 | i1;
```

```
endmodule
module xor2 (input wire i0, i1, output wire o);
  assign o = i0 ^ i1;
endmodule
module nand2 (input wire i0, i1, output wire o);
   wire t;
   and2 and2_0 (i0, i1, t);
   invert invert_0 (t, o);
endmodule
module nor2 (input wire i0, i1, output wire o);
   wire t;
   or2 or2_0 (i0, i1, t);
   invert invert_0 (t, o);
endmodule
module xnor2 (input wire i0, i1, output wire o);
   wire t;
   xor2 xor2_0 (i0, i1, t);
   invert invert_0 (t, o);
endmodule
module and3 (input wire i0, i1, i2, output wire o);
   wire t;
   and2 and2_0 (i0, i1, t);
   and2 and2_1 (i2, t, o);
endmodule
module or3 (input wire i0, i1, i2, output wire o);
   wire t;
   or2 or2_0 (i0, i1, t);
   or2 or2_1 (i2, t, o);
endmodule
module nor3 (input wire i0, i1, i2, output wire o);
   wire t;
   or2 or2_0 (i0, i1, t);
   nor2 nor2_0 (i2, t, o);
endmodule
module nand3 (input wire i0, i1, i2, output wire o);
   wire t;
   and2 and2_0 (i0, i1, t);
   nand2 nand2_1 (i2, t, o);
endmodule
module xor3 (input wire i0, i1, i2, output wire o);
   wire t;
   xor2 xor2_0 (i0, i1, t);
   xor2 xor2_1 (i2, t, o);
endmodule
module xnor3 (input wire i0, i1, i2, output wire o);
   wire t;
   xor2 xor2_0 (i0, i1, t);
   xnor2 xnor2_0 (i2, t, o);
```

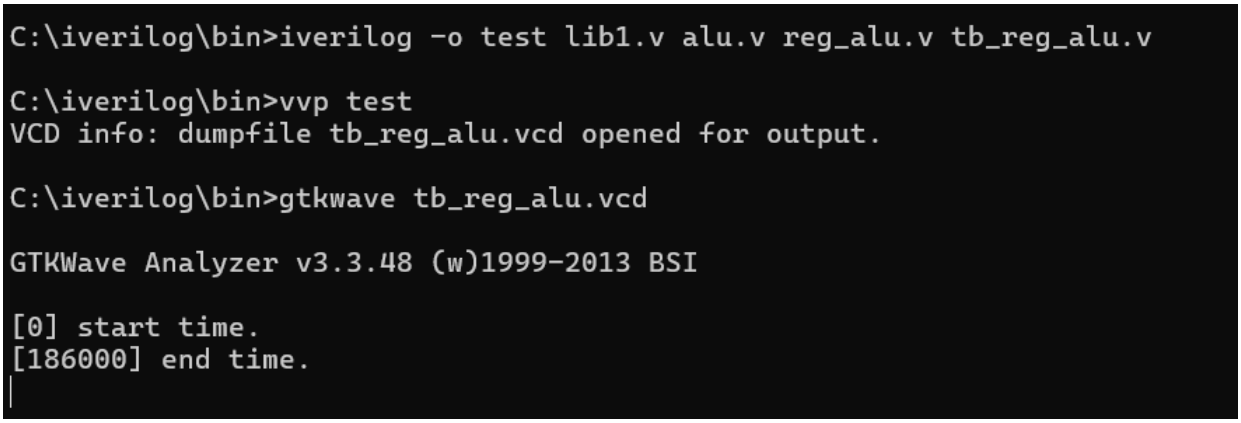
```
endmodule
module mux2 (input wire i0, i1, j, output wire o);
  assign o = (j==0)?i0:i1;
endmodule
module mux4 (input wire [0:3] i, input wire j1, j0, output wire o);
  wire  t0, t1;
  mux2 mux2_0 (i[0], i[1], j1, t0);
  mux2 mux2_1 (i[2], i[3], j1, t1);
mux2 mux2_2 (t0, t1, j0, o);
endmodule
module mux8 (input wire [0:7] i, input wire j2, j1, j0, output wire o);
  wire  t0, t1;
  mux4 mux4_0 (i[0:3], j2, j1, t0);
  mux4 mux4_1 (i[4:7], j2, j1, t1);
mux2 mux2_0 (t0, t1, j0, o);
endmodule
module demux2 (input wire i, j, output wire o0, o1);
  assign o0 = (j==0)?i:1'b0;
  assign o1 = (j==1)?i:1'b0;
endmodule
module demux4 (input wire i, j1, j0, output wire [0:3] o);
  wire  t0, t1;
  demux2 demux2_0 (i, j1, t0, t1);
  demux2 demux2_1 (t0, j0, o[0], o[1]);
  demux2 demux2_2 (t1, j0, o[2], o[3]);
endmodule
module demux8 (input wire i, j2, j1, j0, output wire [0:7] o);
  wire  t0, t1;
  demux2 demux2_0 (i, j2, t0, t1);
demux4 demux4_0 (t0, j1, j0, o[0:3]);
demux4 demux4_1 (t1, j1, j0, o[4:7]);
endmodule
module df (input wire clk, in, output wire out);
  reg df_out;
  always@(posedge clk) df_out <= in;
  assign out = df_out;
endmodule
module dfr (input wire clk, reset, in, output wire out);
  wire reset_, df_in;
  invert invert_0 (reset, reset_);
and2 and2_0 (in, reset_, df_in);
df df_0 (clk, df_in, out);
endmodule
module dfrl (input wire clk, reset, load, in, output wire out);
  wire _in;
  mux2 mux2_0(out, in, load, _in);
dfr dfr_1(clk, reset, _in, out);
endmodule
```

tb_reg_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'o2; 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'o4; test_vecs[7][21:19] = 3'o4;
  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
```

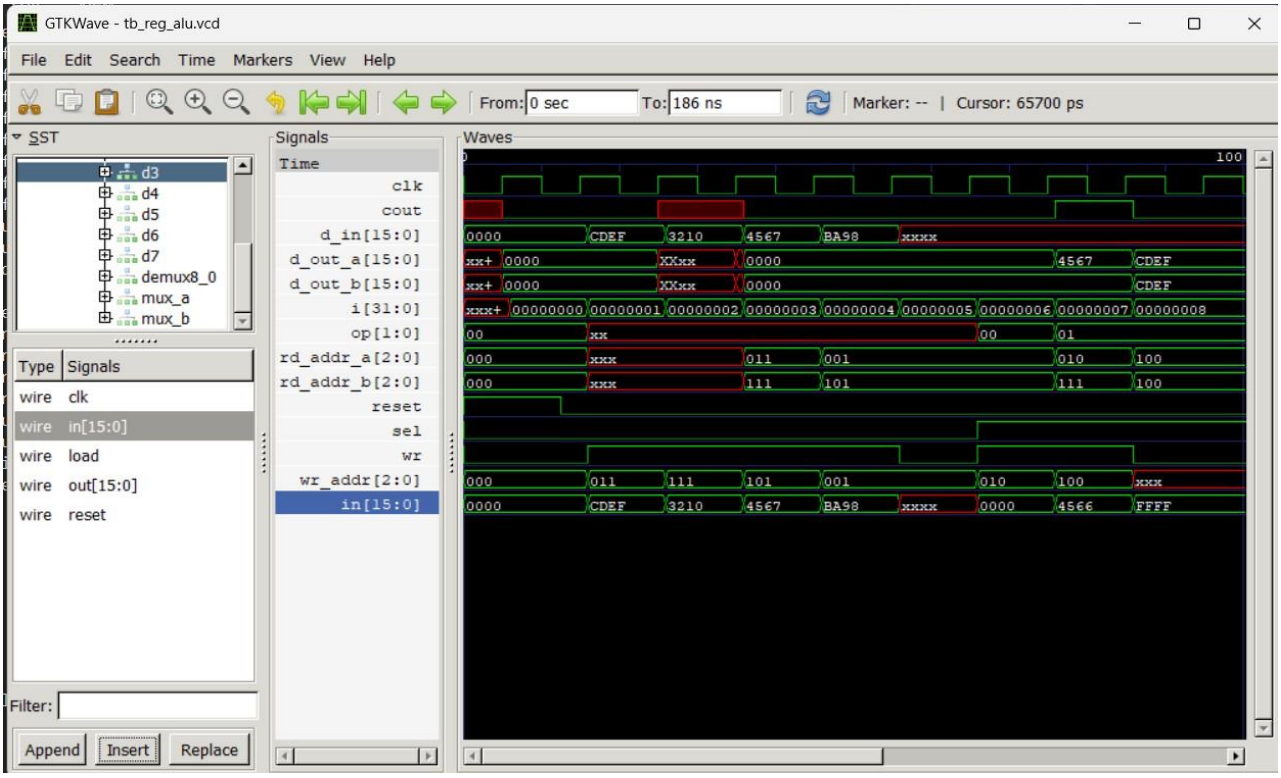
II. Verilog VVP Output Screen Shot



III. GTKWAVE Screenshot

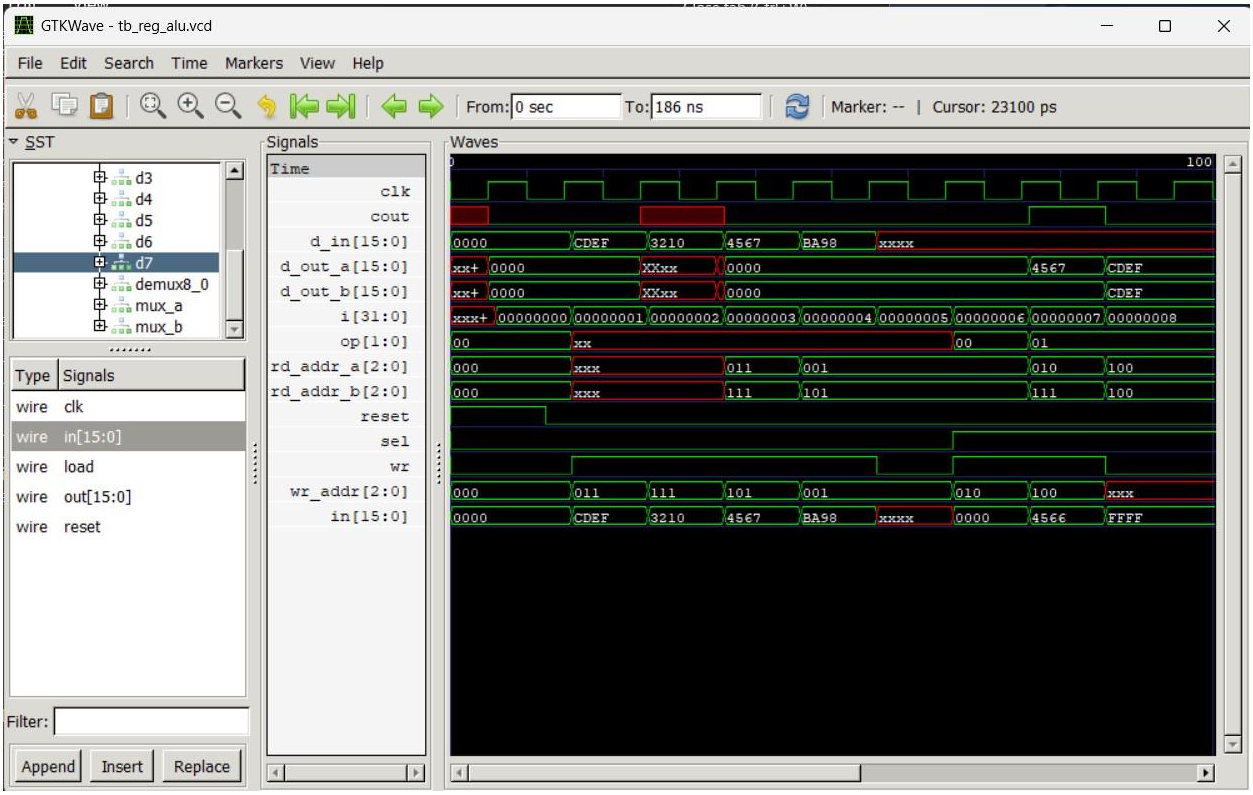
Case:1

when sel is **0**, wr is **1**, wr_addr is **3** (011), and d_in is **CDEF**.



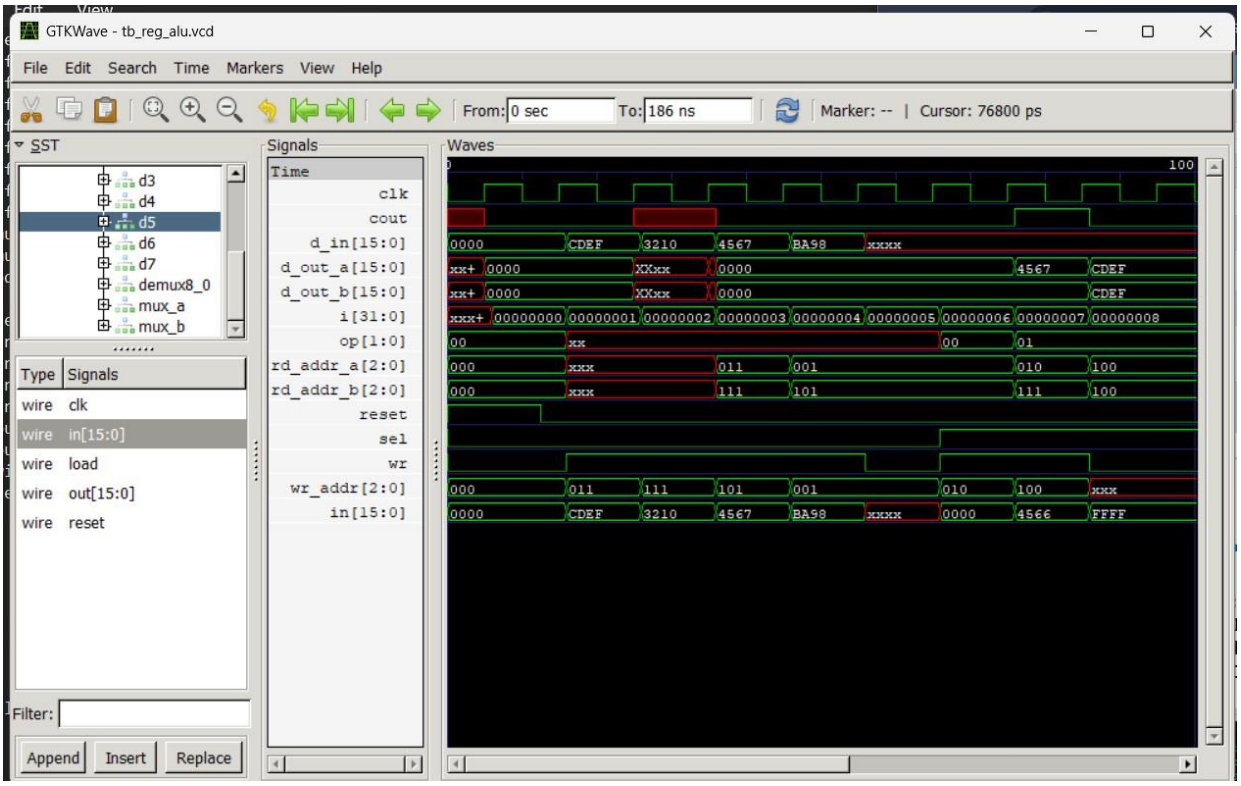
Case:2

when sel is **0**, wr is **1**, wr_addr is **7** (111), and d_in is **3210**.



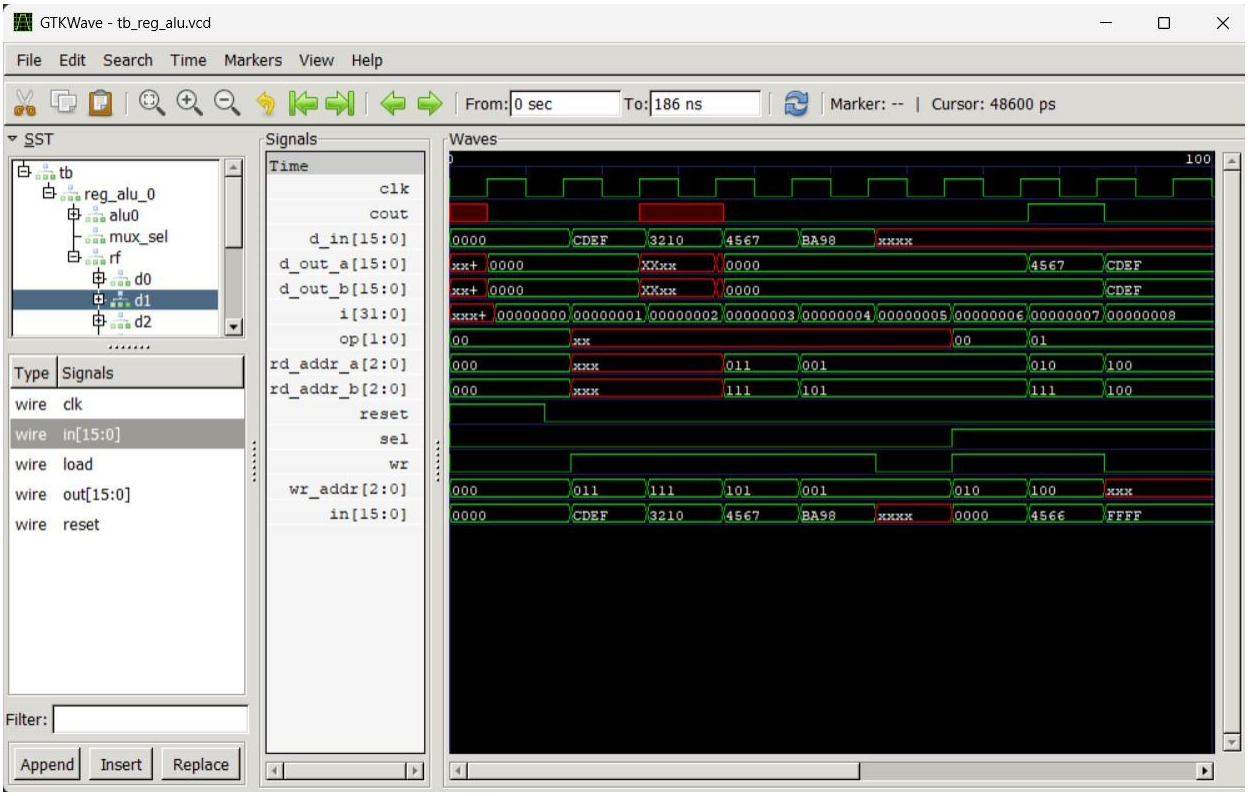
Case:3

when sel is **0**, wr is **1**, wr_addr is **5** (101), and d_in is **4567**.



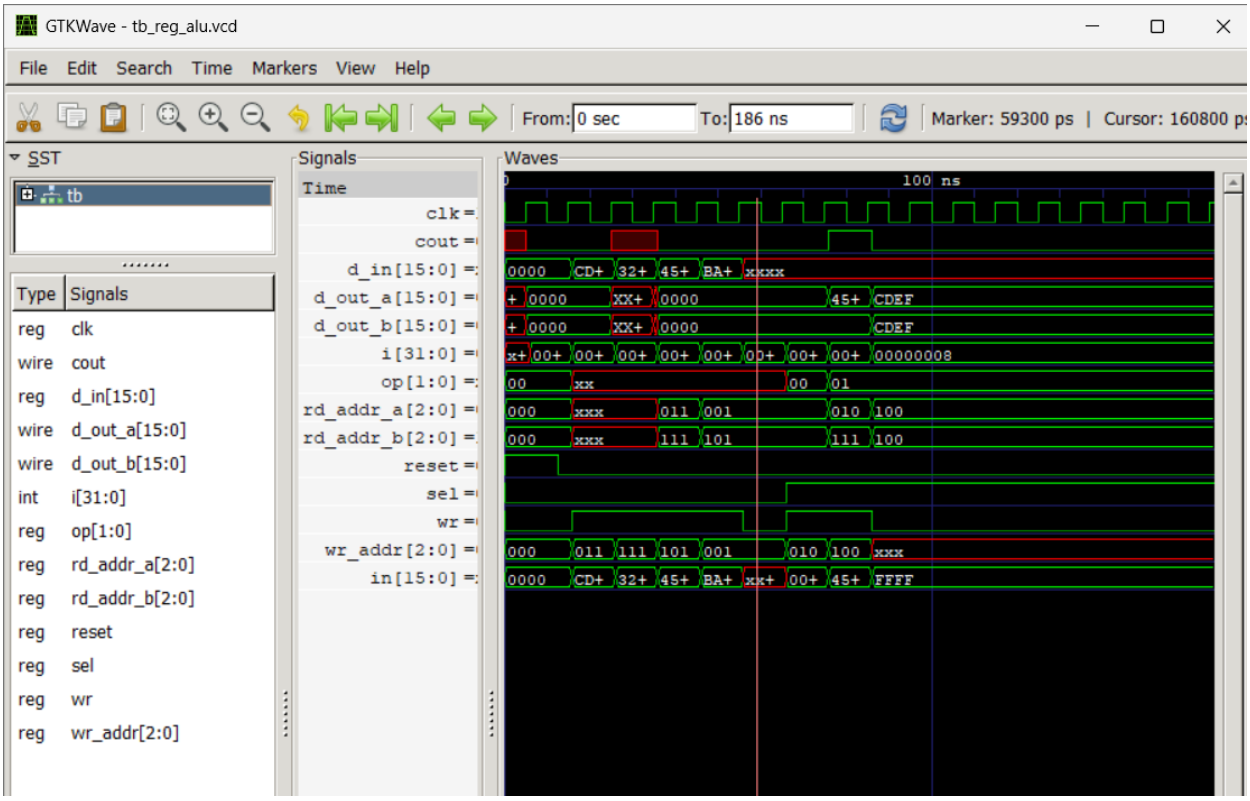
Case:4

when sel is **0**, wr is **1**, wr_addr is **1** (001), and d_in is **BA98**.



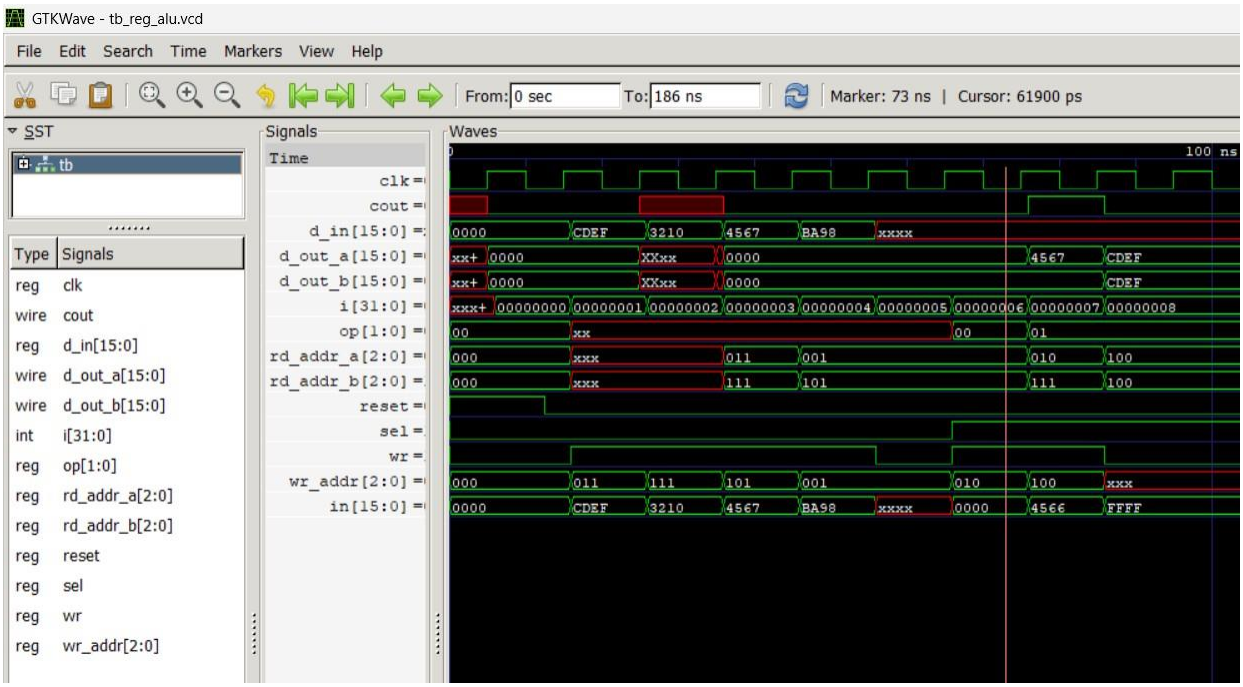
Case:5

when sel is **0**, wr is **0**, rd_addr_a is **1** (001), and rd_addr_b is **5** (101).



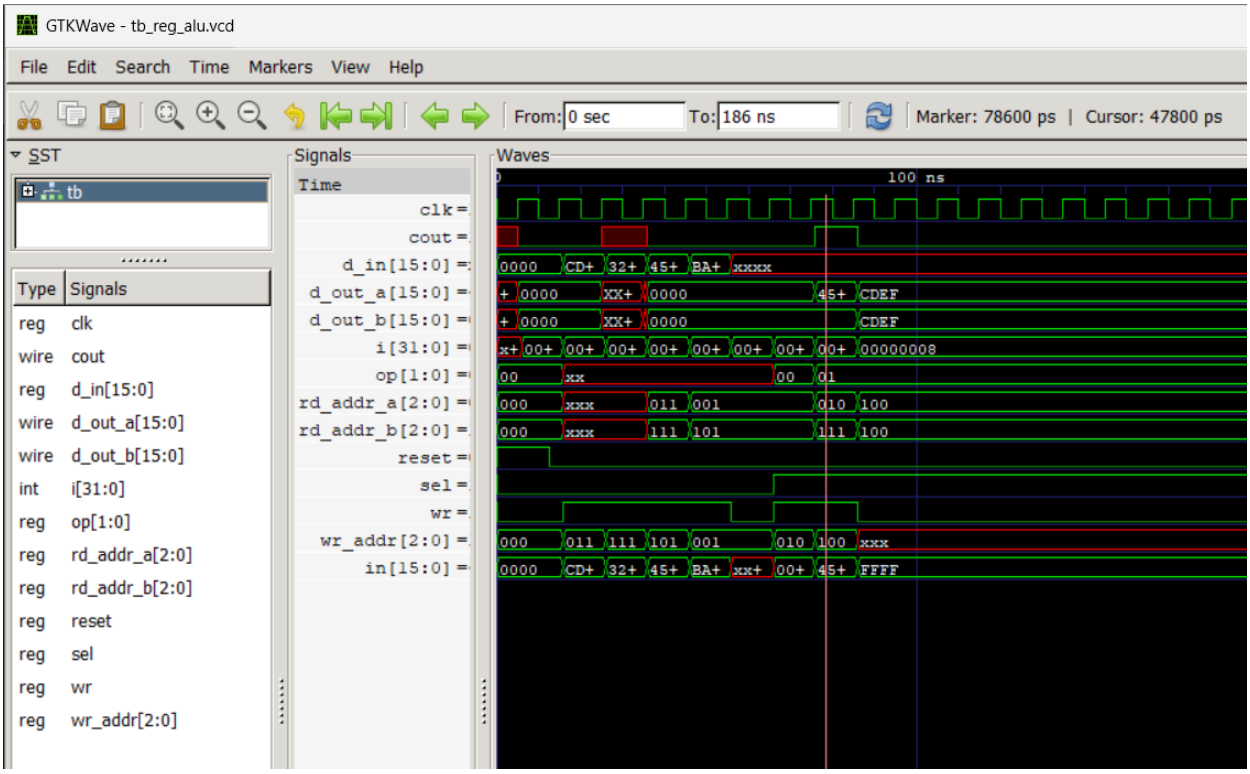
Case:6

when sel is **1**, wr is **1**, op is **00**, rd_addr_a is **1**, rd_addr_b is **5**, and wr_addr is **2** (010).

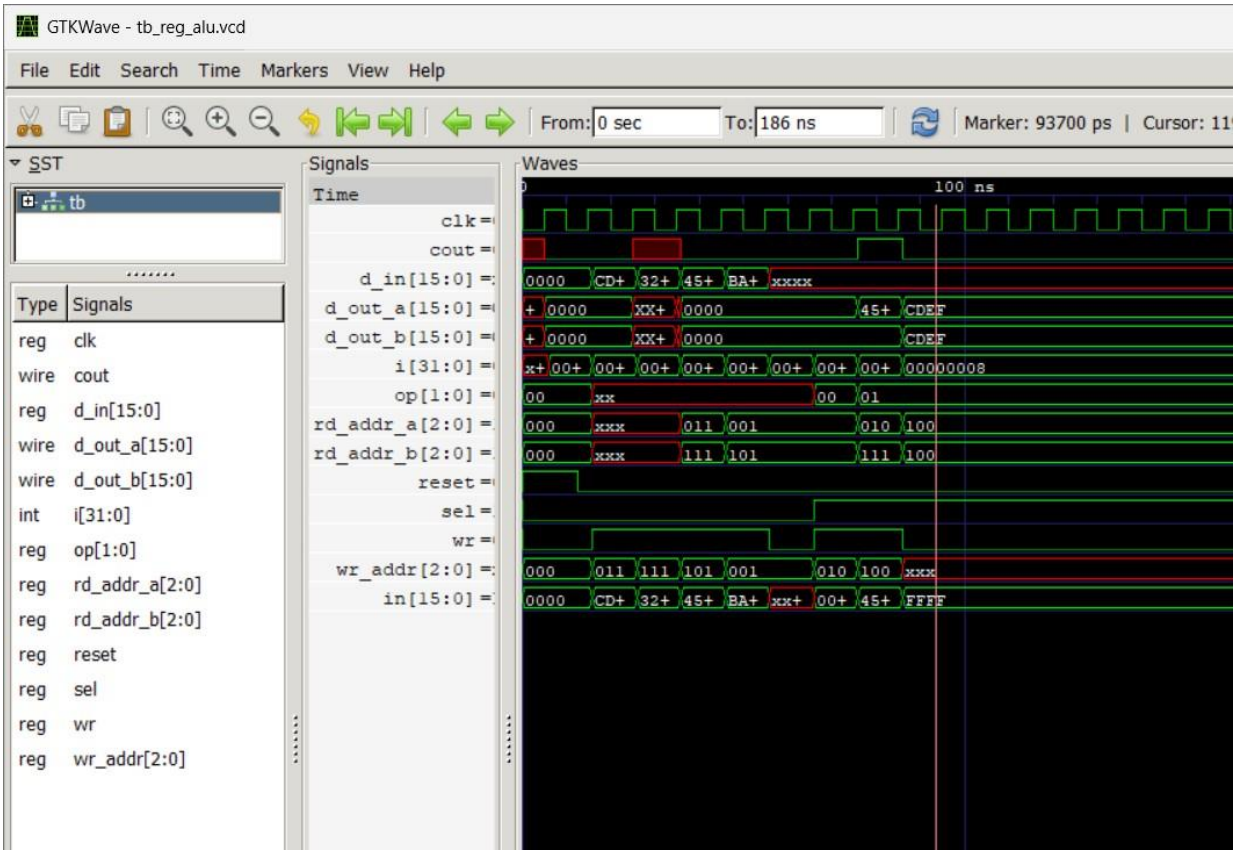


Case:7

when sel is **1**, wr is **1**, op is **01**, rd_addr_a is **2** (010), rd_addr_b is **7** (111), and wr_addr is **4** (100).



Case:8=A time when sel is **1**, wr is **0**, op is **01**, rd_addr_a is **4** (100), and rd_addr_b is **4** (100).



IV. Output Table to be completed and included

sel (28)	wr (27)	op (26-25)	rd_addr_a (24-22)	rd_addr_b (21-19)	wr_addr (18-16)	d_in (15-0)	Output
0	1	xx	x	x	011	CDEF	Reg3 = CDEF
0	1	xx	x	x	111	3210	Reg7 = 3210
0	1	xx	011	111	101	4567	Reg5 = 4567
0	1	xx	001	101	001	BA98	Reg1 = BA98
0	0	xx	001	101	001	xxxx	d_out_a=0000, d_out_b=4567
1	1	00	001	101	010	xxxx	Reg2 = 4567+BA98 = FFFF, cout=0
1	1	01	010	111	100	xxxx	Reg4 = FFFF-3210 = CDEF, cout=1
1	0	01	100	100	x	xxxx	d_out_a - d_out_b = CDEF - CDEF = 0000