Digital Design and Computer Organization Laboratory

UE19CS206

3rd Semester, Academic Year 2020-21

Date:

Name : Achyut Jagini	SRN : PES2UG19CS013	Section : A

Experiment Number: 5 Week # : 5

Title of the Program:

Integration of alu and reg_file to form reg_alu.

Aim of the Program:

To connect the two read outputs and the write input of the of the register file implemented in the previous lab assignment (WEEK 4) to the two inputs and one output of the ALU implemented in the previous assignment(WEEK3)

Code (reg_alu.v)

```
// Write code for modules you need here

module dfrl_16 (input wire  clk, reset, load, input wire [0:15] in, output wire [0:15] out);

 dfrl dfrl_0(clk, reset, load, in[0], out[0]);

 dfrl dfrl_1(clk,reset,load,in[1],out[1]);

dfrl dfrl_2(clk,reset,load,in[2],out[2]);

dfrl dfrl_3(clk,reset,load,in[3],out[3]);

dfrl dfrl_4(clk,reset,load,in[4],out[4]);

dfrl dfrl_5(clk,reset,load,in[5],out[5]);

dfrl dfrl_6(clk,reset,load,in[6],out[6]);

dfrl dfrl_7(clk,reset,load,in[7],out[7]);

dfrl dfrl_8(clk,reset,load,in[8],out[8]);

dfrl dfrl_9(clk,reset,load,in[9],out[9]);

dfrl dfrl_10(clk,reset,load,in[10],out[10]);

dfrl dfrl_11(clk,reset,load,in[11],out[11]);
```

```
26  dfrl dfrl_12(clk,reset,load,in[12],out[12]);
27
28  dfrl dfrl_13(clk,reset,load,in[13],out[13]);
29
30  dfrl dfrl_14(clk,reset,load,in[14],out[14]);
31
32  dfrl dfrl_15(clk,reset,load,in[15],out[15]);
33
34   endmodule
35
36
37  module mux8_16 (input wire [0:15] i0, i1, i2, i3, i4, i5, i6, i7, input wire [0:2] j,
38  output wire [0:15] o);
39
40
41  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]);
42
43  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]);
44
45  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]);
46
47  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]);
48
49  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]);
50
51  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]);
52
53  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]);
54
55  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]);
56
57  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]);
58
```

```
56
57  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]);
58
59   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]);
60
61  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]);
62  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]);
63
64  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]);
65
66  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]);
67
68   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]);
69   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]);
70
71  endmodule
72
73
74
75  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, outpu
76
77
78  // Declare wires here
79
80
81  wire [0:7] load;
82
83  wire [0:15] dout_0, dout_1, dout_2, dout_3, dout_4, dout_5, dout_6, dout_7;
84    dfrl_16 dfrl_16_0(clk, reset, load[0], d_in, dout_0);
85
86   dfrl_16 dfrl_16_1(clk, reset, load[1], d_in, dout_1);
87
```

```
88    dfrl_16 dfrl_16_2(clk, reset, load[2], d_in, dout_2);
89
90     dfrl_16 dfrl_16_3(clk, reset, load[3], d_in, dout_3);
91
92     dfrl_16 dfrl_16_4(clk, reset, load[4], d_in, dout_4);
93
94     dfrl_16 dfrl_16_5(clk, reset, load[5], d_in, dout_5);
95
96     dfrl_16 dfrl_16_6(clk, reset, load[6], d_in, dout_6);
97
98     dfrl_16 dfrl_16_7(clk, reset, load[7], d_in, dout_7);
99
100
101    demux8 demux8_0(wr, wr_addr[2], wr_addr[1], wr_addr[0], load);
102
103   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);
104
105  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);
106  endmodule
107
108
109
110  module mux2_16 (input wire [15:0] i0, i1, input wire j, output wire [15:0] o);
111
112      mux2    mux2_0 (i0[0], i1[0], j,o[0]);
113
114      mux2    mux2_1 (i0[1], i1[1], j, o[1]);
115
116     mux2    mux2_2 (i0[2], i1[2], j, o[2]);
117     mux2    mux2_3 (i0[3], i1[3], j, o[3]);
118     mux2    mux2_4 (i0[4], i1[4], j, o[4]);
119     mux2    mux2_5 (i0[5], i1[5], j, o[5]);
120     mux2    mux2_6 (i0[6], i1[6], j, o[6]);
121     mux2    mux2_7 (i0[7], i1[7], j, o[7]);
```

```
    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 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_alu, d_in_reg;

 wire cout_0;

alu alu_0 (op,d_out_a,d_out_b,d_in_alu,cout_0);

 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);

dfr dfr_0 (clk,reset,cout_0,cout);

endmodule
```

TABLE

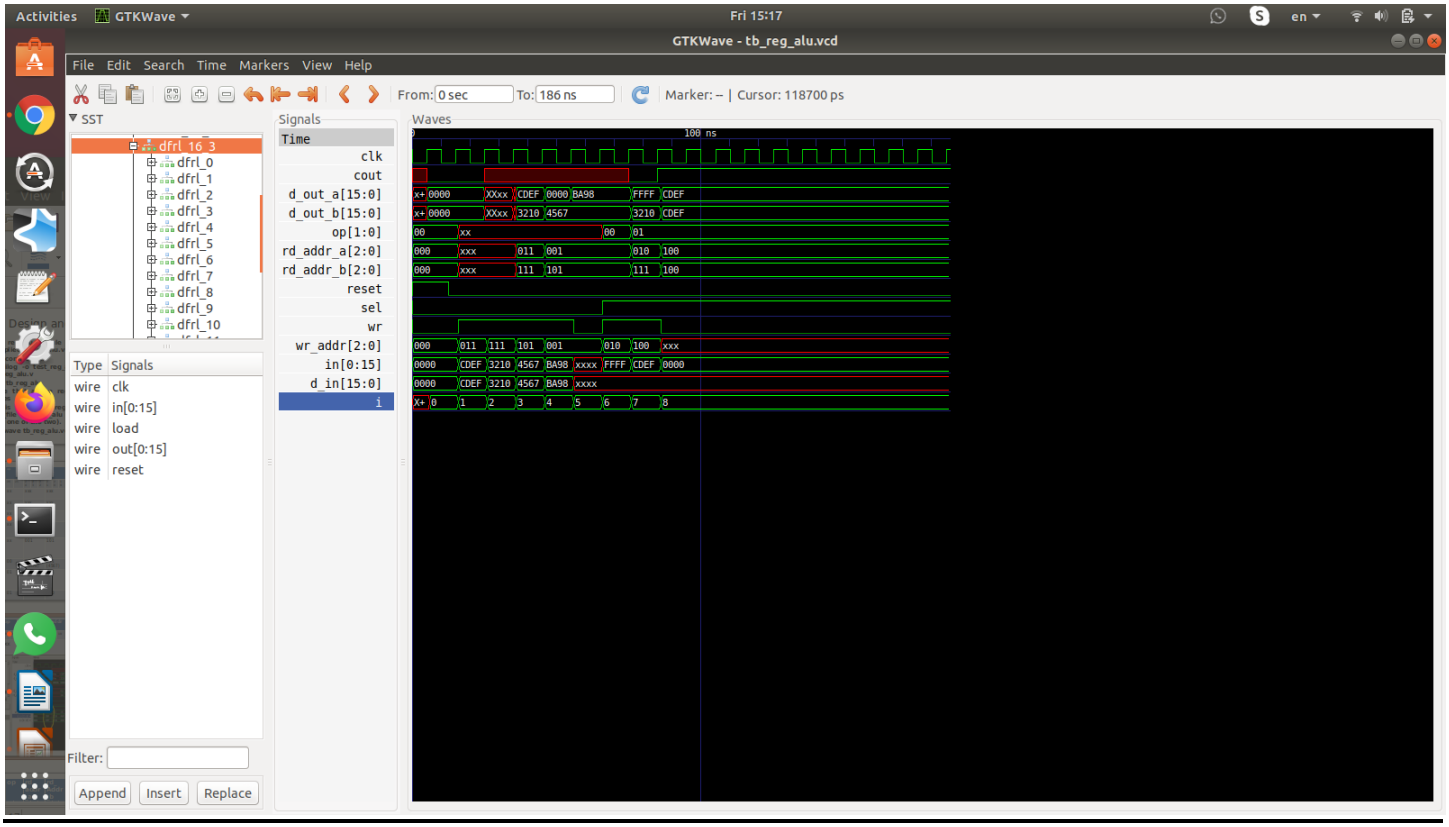
sel	wr	op		rd_ addr_a			rd_ Addr _b			wr_addr			d_in	ALU_output	
28	27	26	25	24	23	22	21	20	19	18	17	16	15-0		
0	1	xx		xxx			xxx			011			CDEF	in[15:0] of REG3= CDEF	

0	1	xx	xxx	xxx	111	3210	in[15:0] of REG7= 3210
0	1	xx	011	111	101	4567	in[15:0] of REG5= 4567
0	1	xx	001	101	001	BA98	in[15:0] of REG1= BA98
0	0	xx	001	101	001	xxxx	out[0:15] of REG 1 =d_out_a=BA98 out[0:15] of REG 5 =d_out_b=4567
1	1	00	001 (BA98)	101 (4567)	010	xxxx	d_in_reg= BA98+4567=FFFF
1	1	01	010	111	100	xxxx	=d_in_reg =d_out_a - d_out_b =FFFF-3210=CDEF
1	0	01	100	100	xxx	xxxx	=d_in_reg =d_out_a - d_out_b =CDEF-CDEF=0000

Output waveform

SCREENSHOT1

CASE1 :WRITE OPERATION

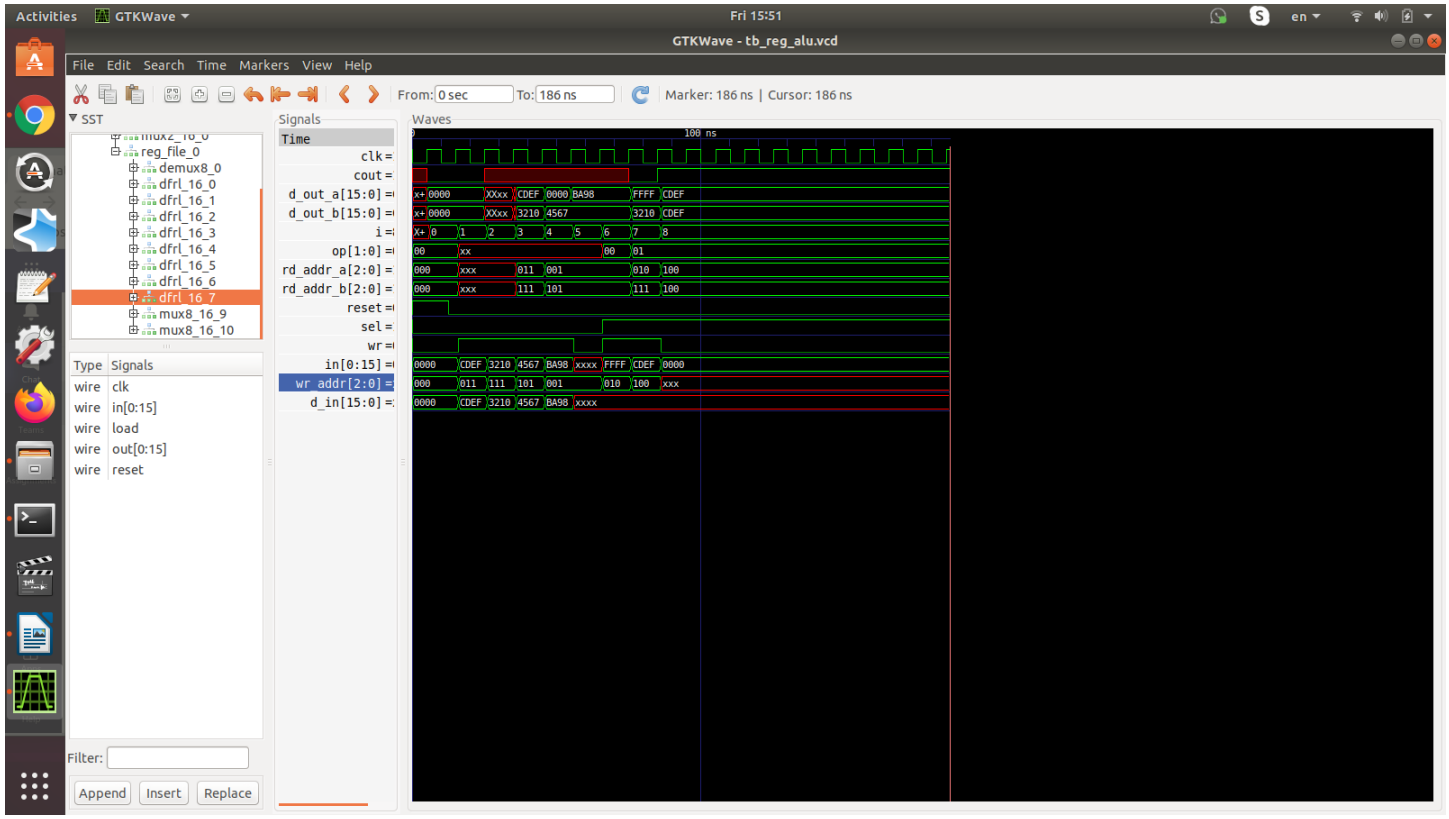


sel	wr	op	rd_addr_a	rd_Addr_b	wr_addr	d_in	ALU output

28	27	26	25	24	23	22	21	20	19	18	17	16	15-0		
0	1	xx		xxx			xxx			011			CDEF	in[15:0] of REG3= CDEF	

SCREENSHOT 2

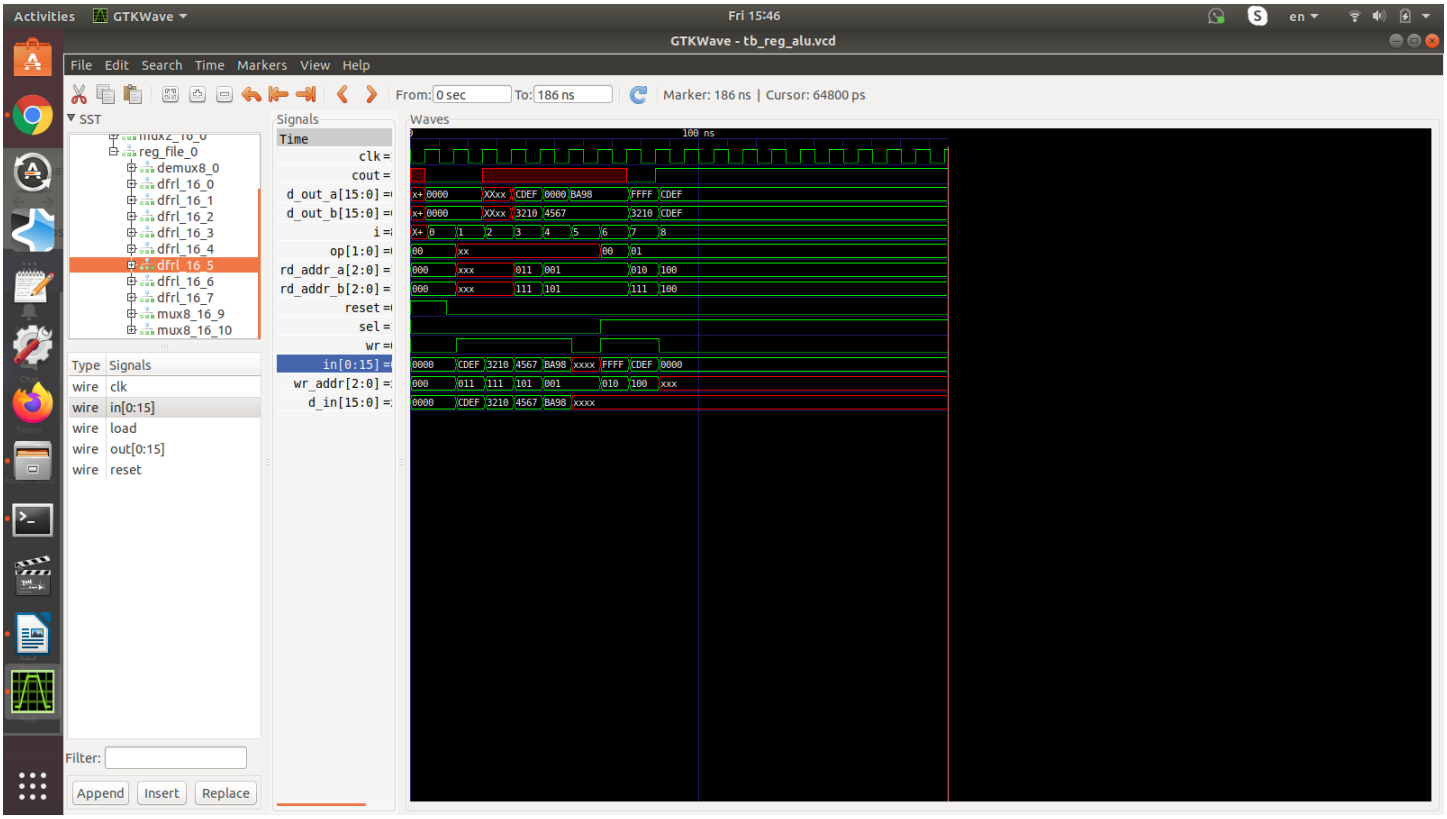
CASE 2 :WRITE OPERATION



sel	wr	op		rd_addr_a			rd_Addr_b			wr_addr			d_in	ALU_output	
28	27	26	25	24	23	22	21	20	19	18	17	16	15-0		
0	1	xx		xxx			xxx			111			3210	in[15:0] of REG7= 3210	

SCREENSHOT 3

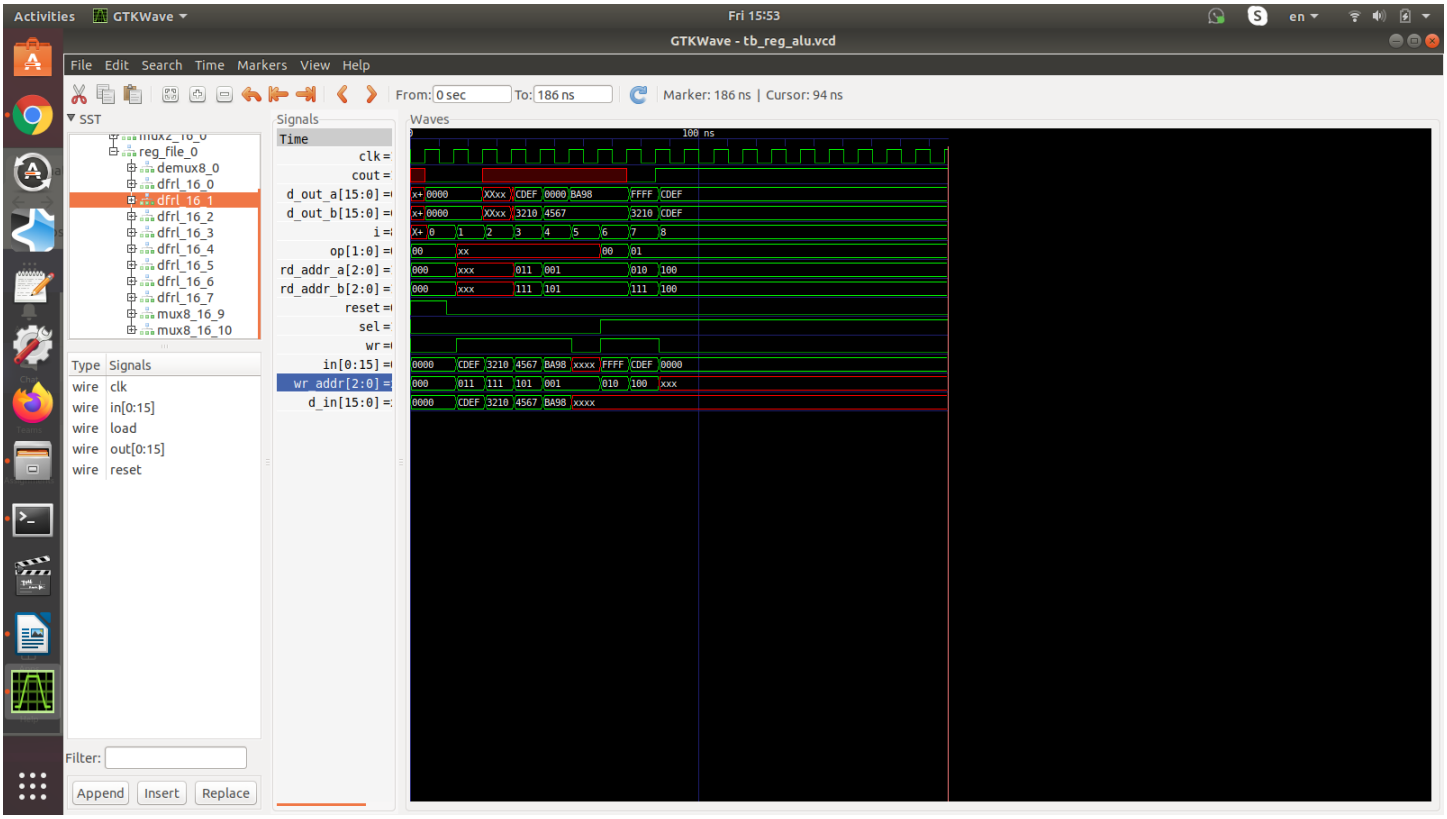
CASE 3: WRITE OPERATION



sel	wr	op		rd_addr_a			rd_Addr_b			wr_addr			d_in	ALU output	
28	27	26	25	24	23	22	21	20	19	18	17	16	15-0		
0	1	xx		011			111			101			4567	in[15:0] of REG5= 4567	

SCREENSHOT 4

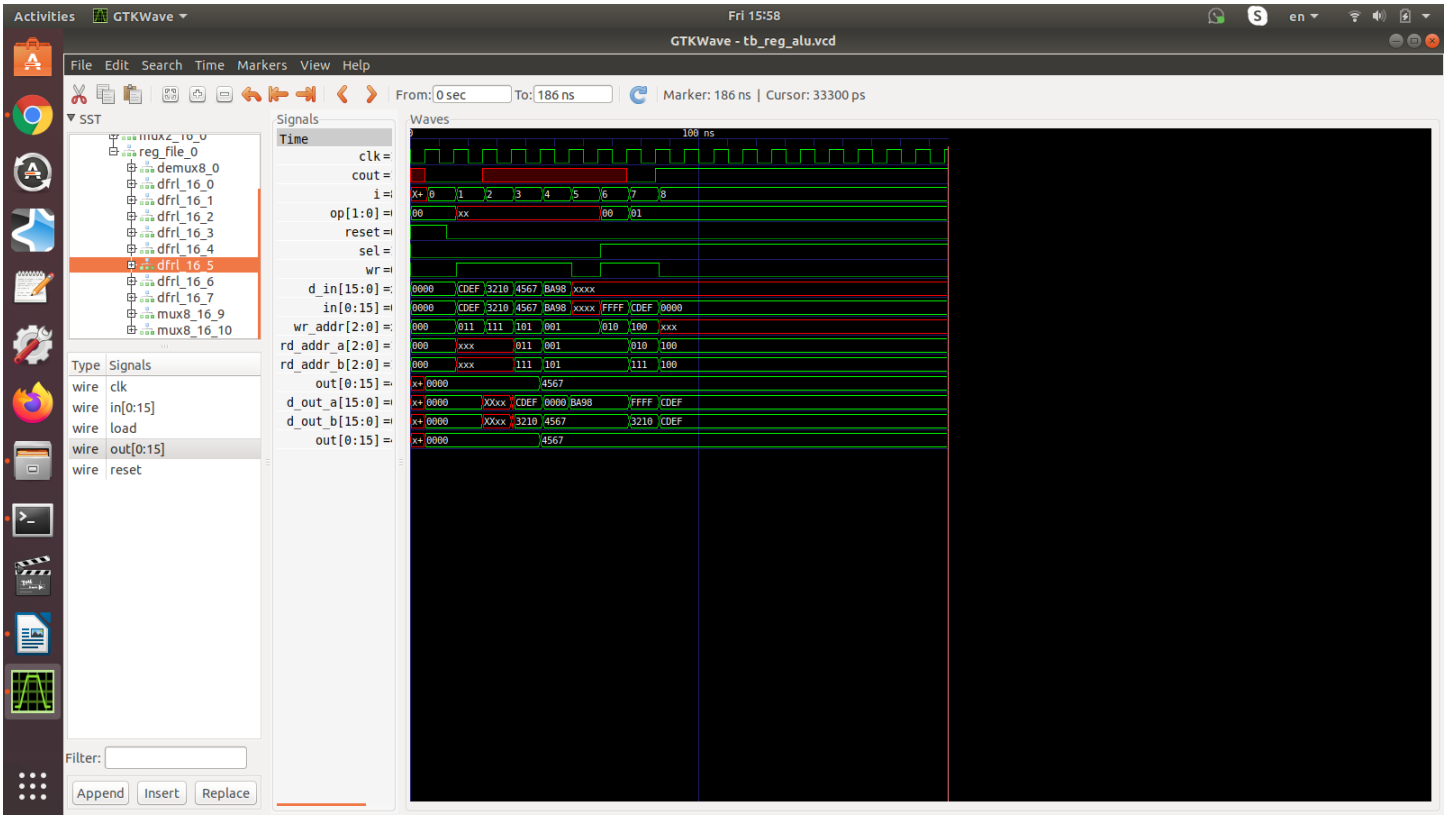
CASE 4: WRITE OPERATION



sel	wr	op		rd_addr_a			rd_Addr_b			wr_addr			d_in	ALU output	
28	27	26	25	24	23	22	21	20	19	18	17	16	15-0		
0	1	xx		001			101			001			BA98	in[15:0] of REG1= BA98	

SCREENSHOT 5

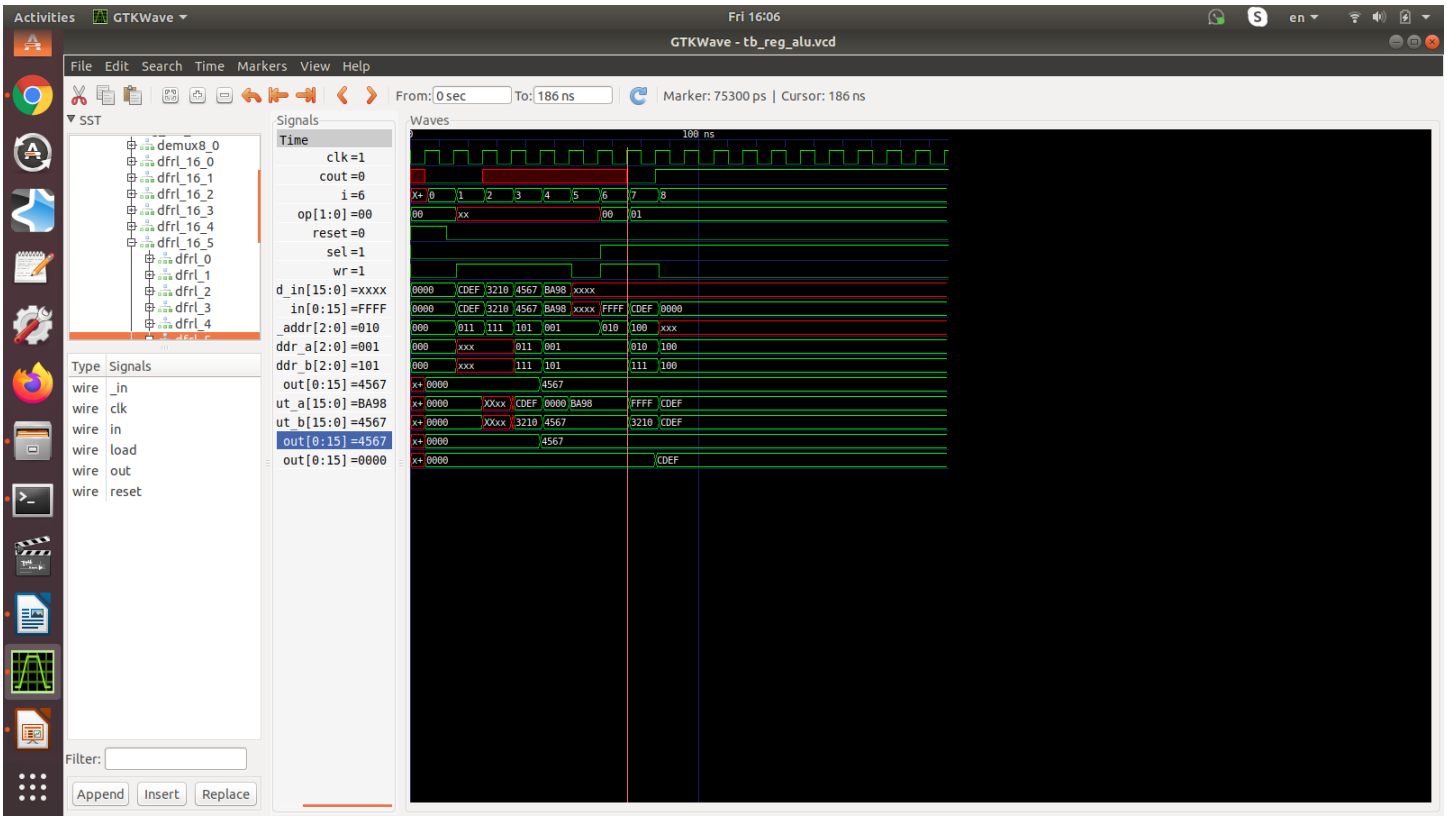
CASE 5: READ OPERATION



sel	wr	op		rd_addr_a			rd_Addr_b			wr_addr			d_in	ALU output	
28	27	26	25	24	23	22	21	20	19	18	17	16	15-0		
0	0	xx		001			101			001			xxxx	out[0:15] of REG 1 =d_out_a=BA98 out[0:15] of REG 5 =d_out_b=4567	

SCREENSHOT 6

CASE 6: READ OPERATION



sel	wr	op		rd_addr_a			rd_Addr_b			wr_addr			d_in	ALU output	
28	27	26	25	24	23	22	21	20	19	18	17	16	15-0		
1	0	01		100			100			xxx			xxxx	=d_in_reg =d_out_a - d_out_b =CDEF-CDEF=0000	