Digital Design and Computer Organization Laboratory

# UE20CS206

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

Date:

| | | |
|---|---|---|
| Name:YAMAN GUPTA | SRN: PES2UG21CS619 | Section J |

Week#__5_________

Program Number :___1____

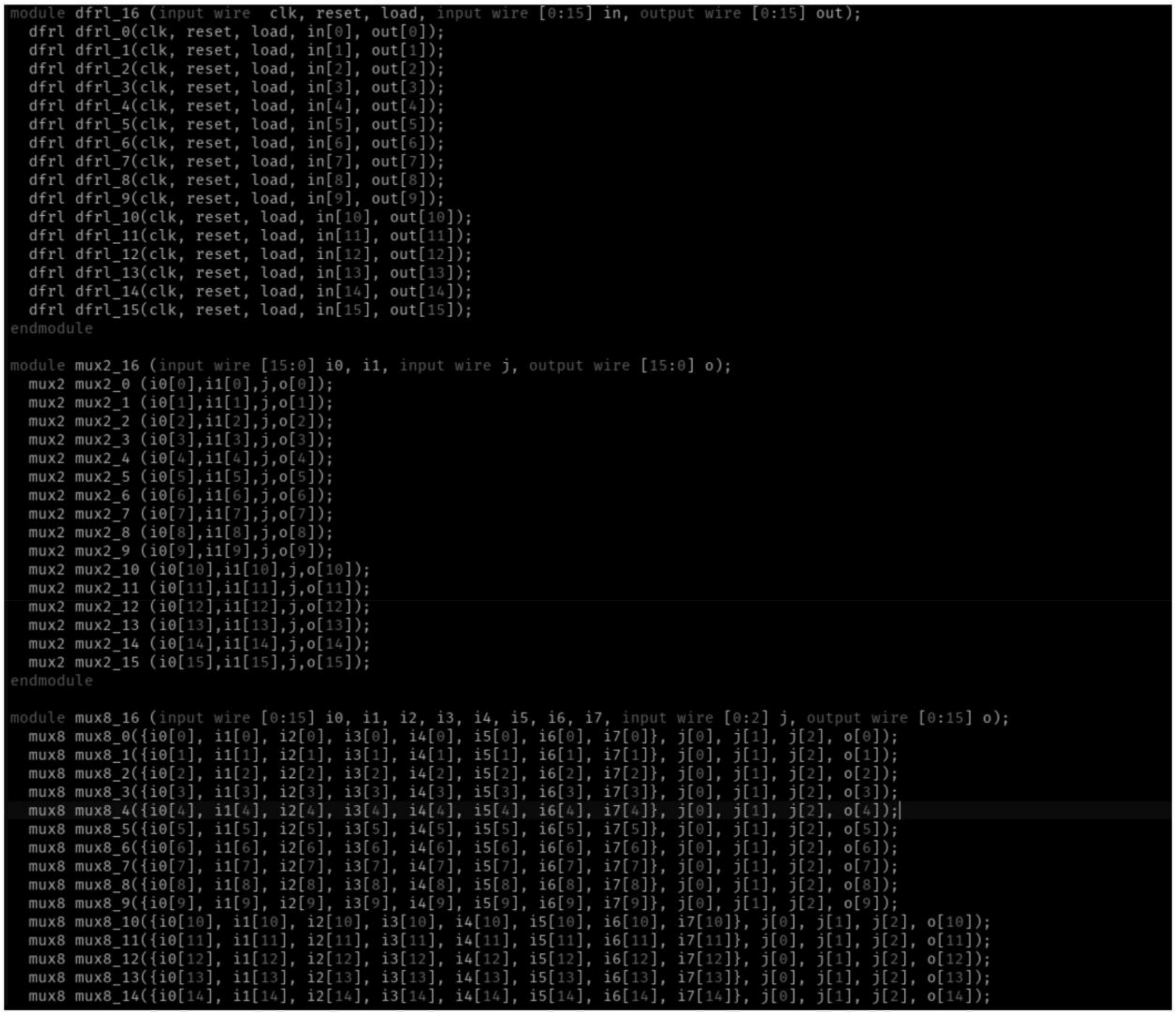
Title of the Program

**REGISTER FILE**

Aim:

**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.**

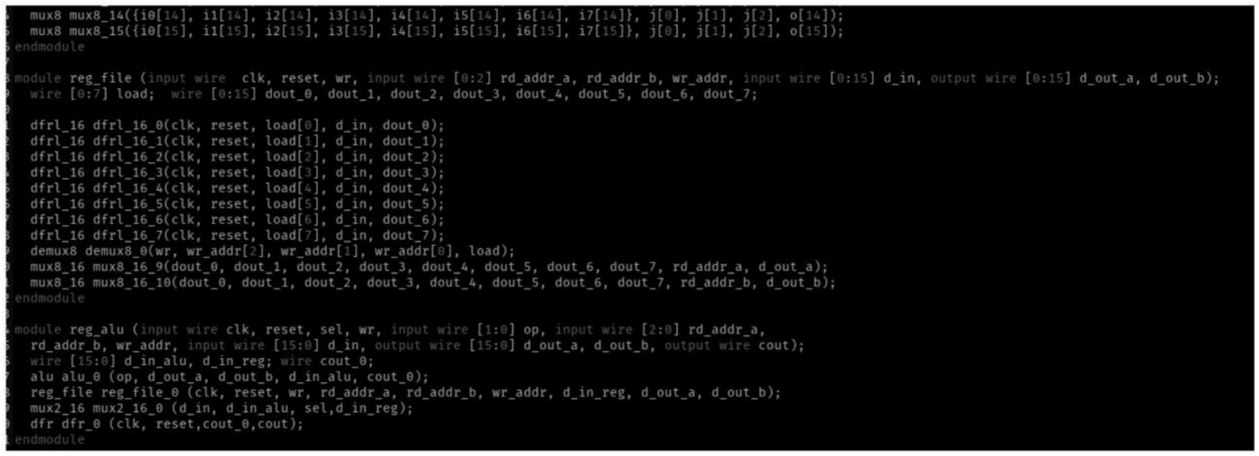
1: Paste the Screen Shot of the source code in reg_alu.v



```
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]);
  dfrl dfrl_12(clk, reset, load, in[12], out[12]);
  dfrl dfrl_13(clk, reset, load, in[13], out[13]);
  dfrl dfrl_14(clk, reset, load, in[14], out[14]);
  dfrl dfrl_15(clk, reset, load, in[15], out[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 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_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 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;

  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);
  demux8 demux8_0(wr, wr_addr[2], wr_addr[1], wr_addr[0], load);
  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 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] 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
```

## 2. Complete the truth table for all the 8 rows

| **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 | CDEF | Reg3=CDEF |
| 0 | 1 | xx | x | x | x | x | x | x | 1 | 1 | 1 | 3210 | Reg7= 3210 |
| 0 | 1 | xx | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 4567 | Reg5=4567 |
| 0 | 1 | xx | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | BA98 | Reg1=BA98 |
| 0 | 0 | xx | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | xxxx | d_out_a=BA98 d_out_b=4567 |
| 1 | 1 | 00 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | xxxx | Reg2=4567+BA98 |

=FFFF, cout =0

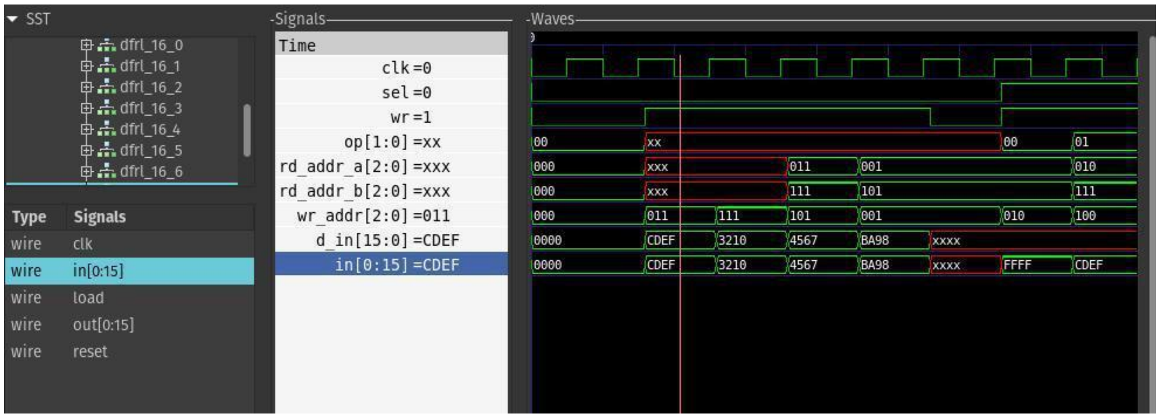
| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 01 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | xxxx | Reg4 = FFFF-3210=CDEF, cout=1 |
| 1 | 0 | 01 | 1 | 0 | 0 | 1 | 0 | 0 | x | x | x | xxxx | d_out_a-d_out_b= =CDEF-CDEF=0000 |

## 3: Paste the Screen shot of the GTKWave form

### I. SCREENSHOT1

**CASE1 (Write operation):**

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

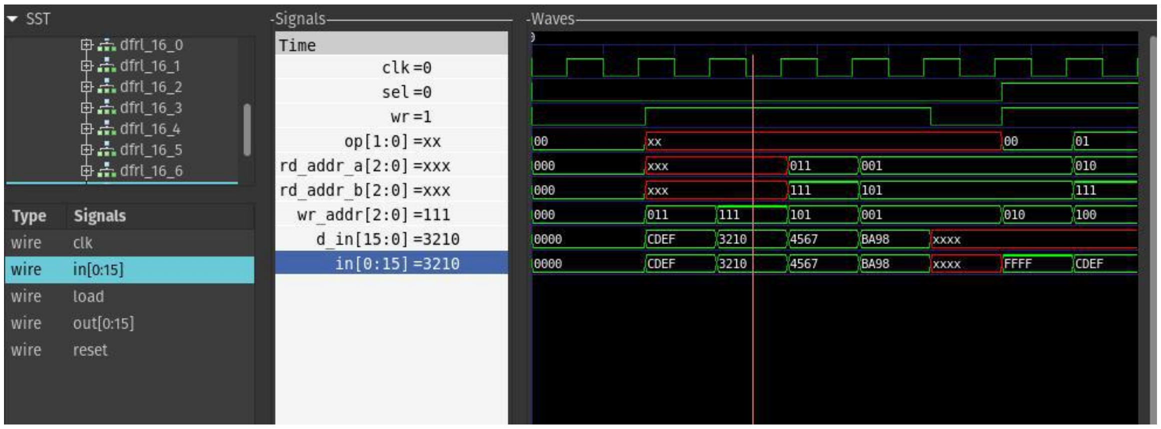


```
Time
clk=0
sel=0
wr=1
op[1:0]=xx
rd_addr_a[2:0]=xxx
rd_addr_b[2:0]=xxx
wr_addr[2:0]=011
d_in[15:0]=CDEF
in[0:15]=CDEF
```

### II. SCREENSHOT 2

**CASE2 (Write operation):**

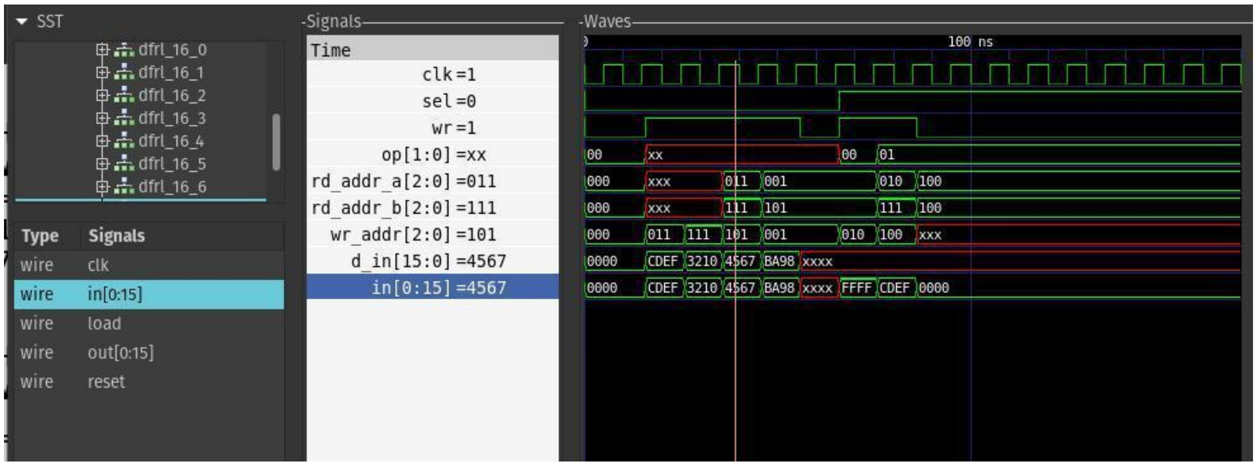
sel =0,wr=1 ,Write Address=111,d_in=3210,
Verify in[15:0] of Register 7



## III. SCREENSHOT 3

### **CASE 3 (Write operation):**

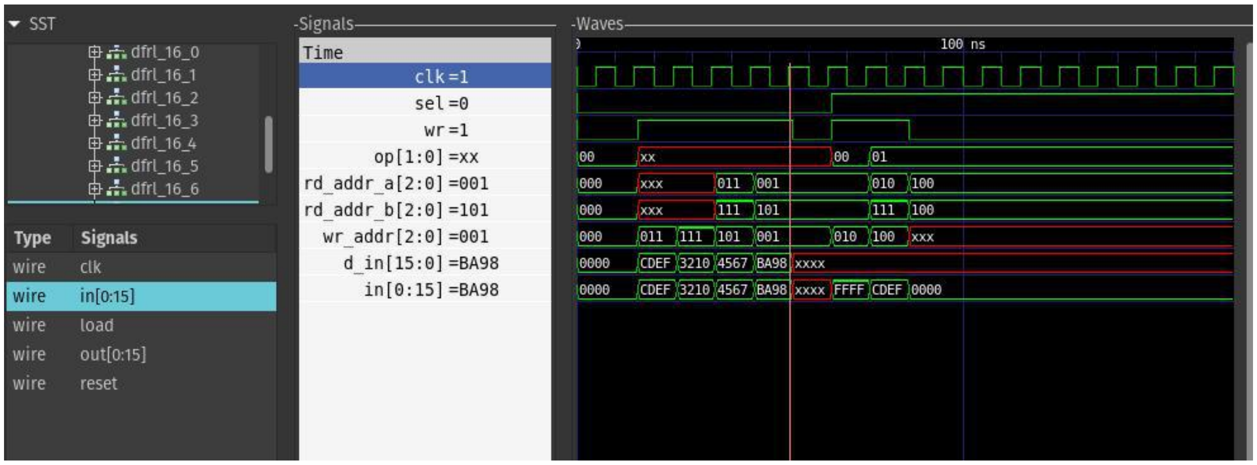
sel =0,wr=1 ,rd_addr_a=011, rd_addr_b=111,wr_addr=101, d_in=4567,Verify in[15:0] of Register 5



## IV. SCREENSHOT 4

### **CASE 4 (Write operation):**

sel =0,wr=1 , rd_addr_a=001, rd_addr_b=101,wr_addr=001, d_in=BA98,Verify in[15:0] of Register 1

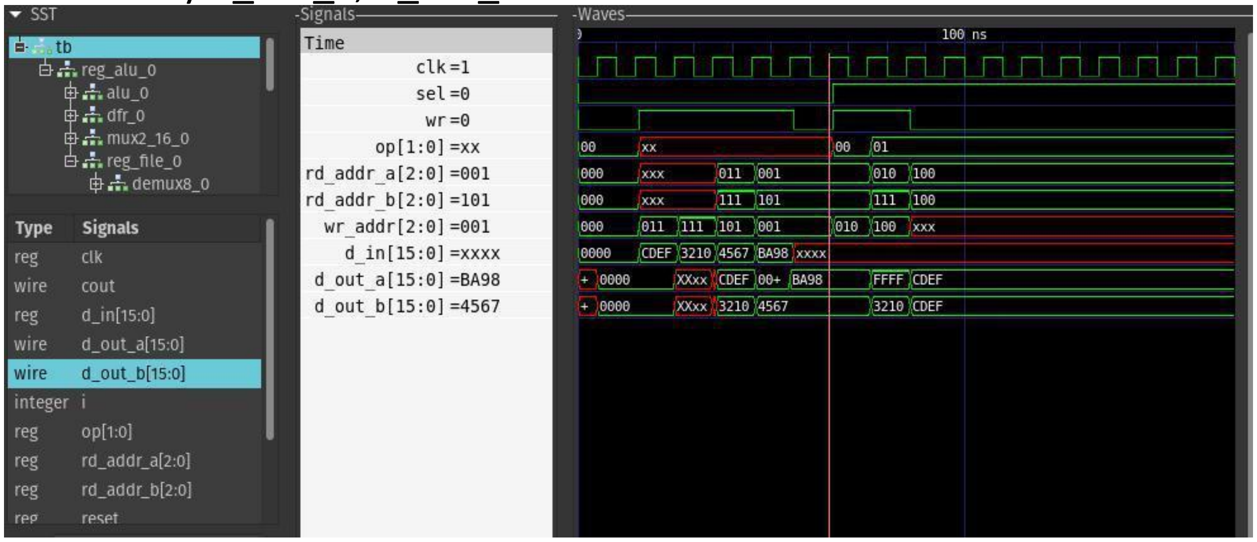


```
Time
clk=1
sel=0
wr=1
op[1:0]=xx
rd_addr_a[2:0]=001
rd_addr_b[2:0]=101
wr_addr[2:0]=001
d_in[15:0]=BA98
in[0:15]=BA98
```

```
op[1:0]         00 xx 00 01
rd_addr_a[2:0]  000 xxx 011 001 010 100
rd_addr_b[2:0]  000 xxx 111 101 111 100
wr_addr[2:0]    000 011 111 101 001 010 100 xxx
d_in[15:0]      0000 CDEF 3210 4567 BA98 xxxx
in[0:15]        0000 CDEF 3210 4567 BA98 xxxx FFFF CDEF 0000
```

V. SCREENSHOT 5 **CASE 5 (Read operation):**

sel =0,wr=0 , rd_addr_a=001, rd_addr_b=101,wr_addr=001, Verify d_out_a, d_out_b

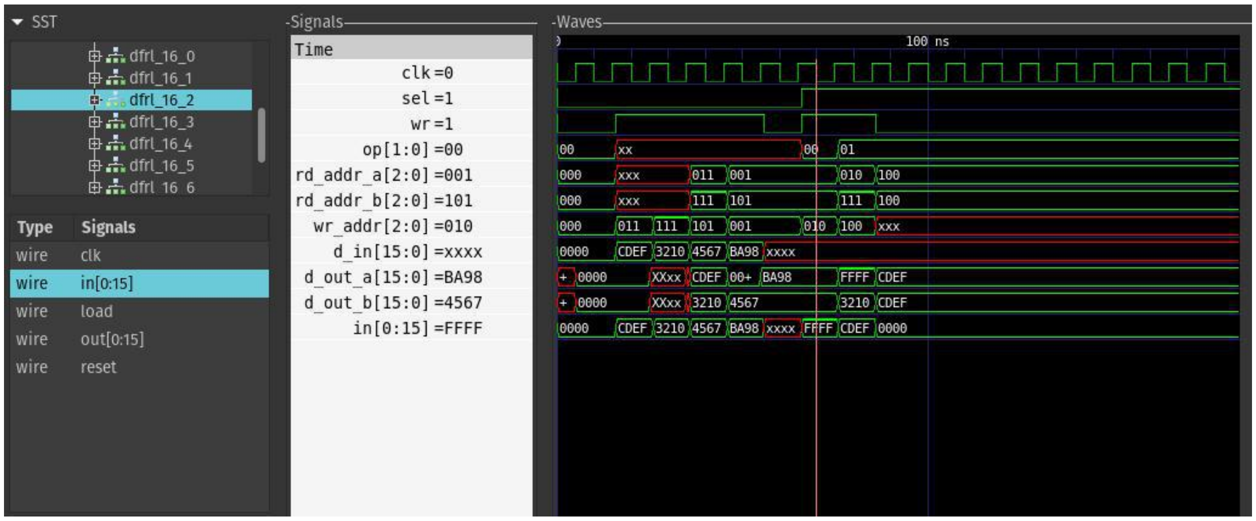


```
Time
clk=1
sel=0
wr=0
op[1:0]=xx
rd_addr_a[2:0]=001
rd_addr_b[2:0]=101
wr_addr[2:0]=001
d_in[15:0]=xxxx
d_out_a[15:0]=BA98
d_out_b[15:0]=4567
```

```
op[1:0]         00 xx 00 01
rd_addr_a[2:0]  000 xxx 011 001 010 100
rd_addr_b[2:0]  000 xxx 111 101 111 100
wr_addr[2:0]    000 011 111 101 001 010 100 xxx
d_in[15:0]      0000 CDEF 3210 4567 BA98 xxxx
d_out_a[15:0]   + 0000 XXxx CDEF 00+ BA98 FFFF CDEF
d_out_b[15:0]   + 0000 XXxx 3210 4567 3210 CDEF
```

VI. SCREENSHOT 6 **CASE 6 (Write operation after addition):**

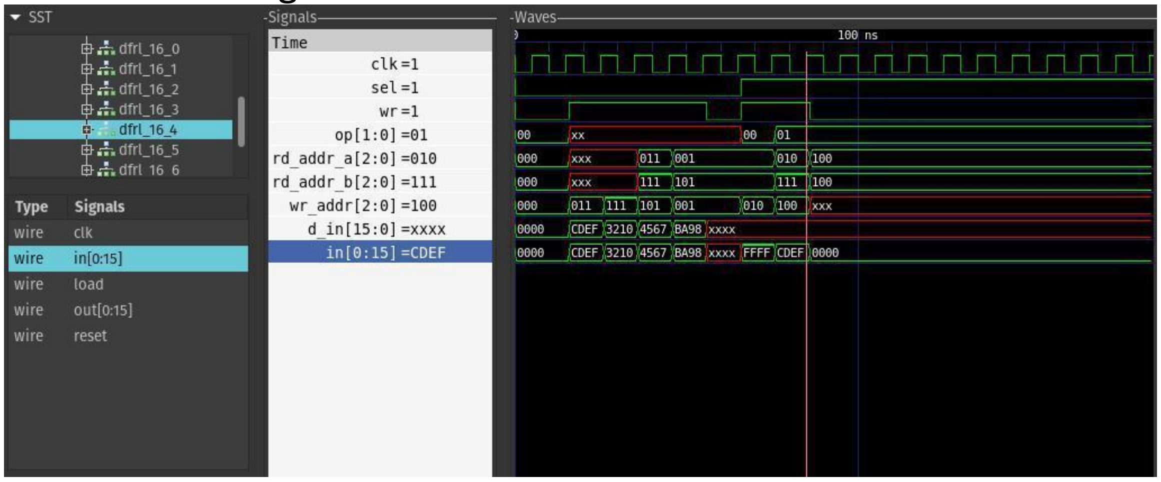
sel =1,wr=1,op=00, rd_addr_a=001, rd_addr_b=101,wr_addr= 010 d_in = XXXX,ALU output=d_out_a + d_out_b=4567+BA98=FFFF to be written to Reg2



```
Time
clk=0
sel=1
wr=1
op[1:0]=00
rd_addr_a[2:0]=001
rd_addr_b[2:0]=101
wr_addr[2:0]=010
d_in[15:0]=xxxx
d_out_a[15:0]=BA98
d_out_b[15:0]=4567
in[0:15]=FFFF
```

VII. SCREENSHOT 7 **CASE 7(Write operation after subtraction):**

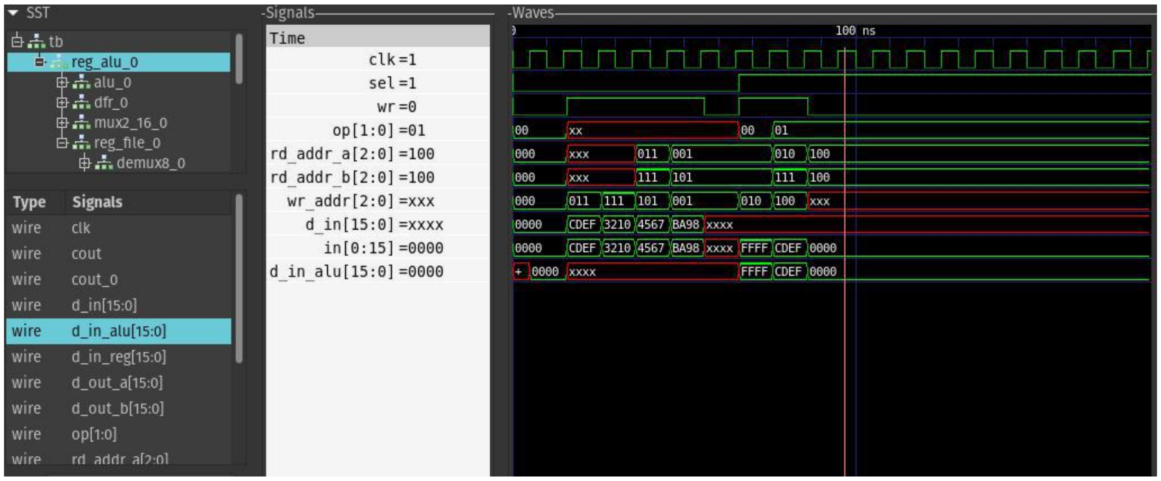
sel =1,wr=1,op=01, rd_addr_a=010, rd_addr_b=111,wr_addr= 100 d_in = XXXX,ALU output=Reg2-Reg7=FFFF-3210=CDEF to be written to Reg4



```
Time
clk=1
sel=1
wr=1
op[1:0]=01
rd_addr_a[2:0]=010
rd_addr_b[2:0]=111
wr_addr[2:0]=100
d_in[15:0]=xxxx
in[0:15]=CDEF
```

VIII. SCREENSHOT 8 **CASE 8(Write operation after subtraction):**

sel =1,wr=0,op=01, rd_addr_a=100, rd_addr_b=100,wr_addr= xxx d_in = XXXX,ALU output=Reg4-Reg4=CDEF-CDEF =0000



```
tb
  reg_alu_0
    alu_0
    dfr_0
    mux2_16_0
    reg_file_0
      demux8_0
```

| Type | Signals |
|---|---|
| wire | clk |
| wire | cout |
| wire | cout_0 |
| wire | d_in[15:0] |
| wire | d_in_alu[15:0] |
| wire | d_in_reg[15:0] |
| wire | d_out_a[15:0] |
| wire | d_out_b[15:0] |
| wire | op[1:0] |
| wire | rd_addr_a[2:0] |

```
Time
          clk =1
          sel =1
           wr =0
     op[1:0] =01
rd_addr_a[2:0] =100
rd_addr_b[2:0] =100
  wr_addr[2:0] =xxx
   d_in[15:0] =xxxx
     in[0:15] =0000
d_in_alu[15:0] =0000
```

**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:YAMAN GUPTA

Name:YAMAN GUPTA

SRN: PES2UG21CS619

Section: J

Date: OCT 7TH,22