CSE-331/HW2

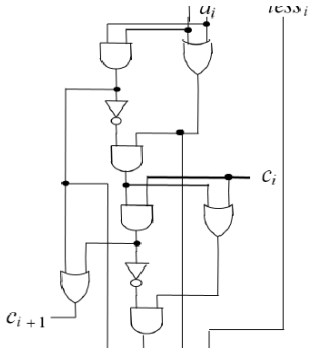
My desing include

- 1 bit ALU → module alu1(out,set,cout,a,b,cin,less,op);

```
not (not1, b);
not (not2, op[2]);
and (ww,b,not2);
and (w,not1,op[2]);
or (xor1,ww,w);
```

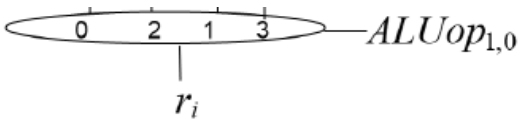
b and op[2] input for xor operation and xor1 assigned output

```
and (and1, a, xor1);
or (or1, a, xor1);
not (nand1 ,and1);
and (and2 , nand1, or1
and (and3 ,and2 ,cin);
or (or2 , and2, cin);
not (nand3, and3);
and (set, nand3, or2);
or (cout, and1 ,and3);
```



Full adder part

```
mux1 hadi(out,and1,or1,set,less,op[1:0]);
```



Less,carry-out,set,carry-in mux

- 4X1 MUX → module mux1 (out,c0,c1,c2,c3,s1);

Select input c0,c1,c2,c3

operatian select s1,

- 32 bit ALU → module orkun(zero, out, overflow, cout, a, b, op,asd,dsa,less);

```
alu1
    a0(out[0], asd[0], dsa[0], a[0], b[0], op[2], less, op),
    a1(out[1], asd[1], dsa[1], a[1], b[1], dsa[0], 0, op),
    a2(out[2], asd[2], dsa[2], a[2], b[2], dsa[1], 0, op),
    a3(out[3], asd[3], dsa[3], a[3], b[3], dsa[2], 0, op),
    a4(out[4], asd[4], dsa[4], a[4], b[4], dsa[3], 0, op),
    a5(out[5], asd[5], dsa[5], a[5], b[5], dsa[4], 0, op),
    a6(out[6], asd[6], dsa[6], a[6], b[6], dsa[5], 0, op),

    …
```

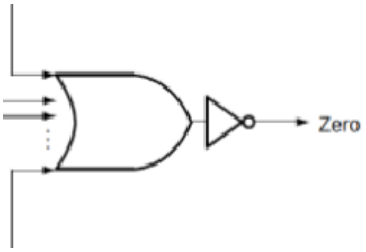
a , b, op, less input

p[2:0]: 3-bit opcode for selection operation.

İf Less is difference 0 ,depens on last ALU

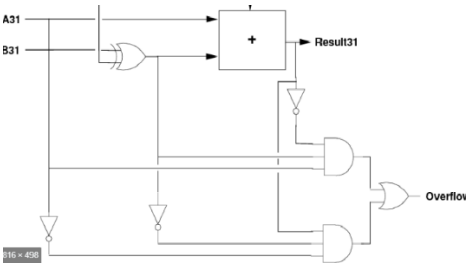
Example a1 ,dsa[1], cout dsa[0]

Occur 32 bit ALU



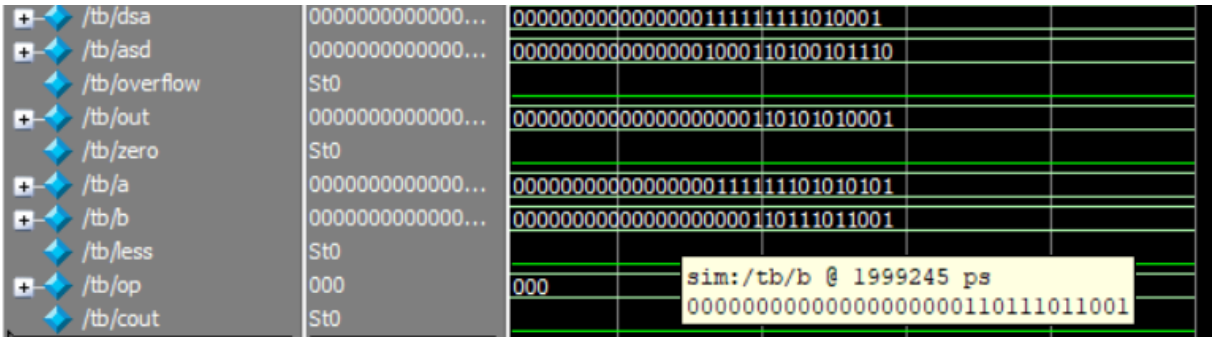

```
or (zeroOrOut, out[0], out[1], out[2], out[3], out[4], out[5], out[6],
not z1(zero, zeroOrOut);
```

all the bits of the result are OR'd to see if any are non-zero




```
not (notCin, dsa[30]);
not (notCout, cout);
and (and1 , notCin , cout);
and (and2, dsa[30],notCout);
or (overflow ,and1, and2);
```

MODELSİM

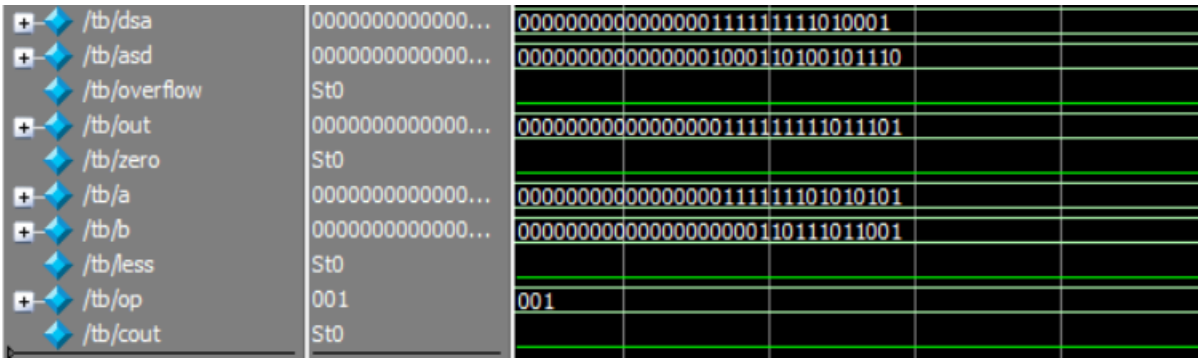


a=00000000000000000111111101010101

b=00000000000000000000110111011001

op=000 ->and operation

out=00000000000000000000110101010001

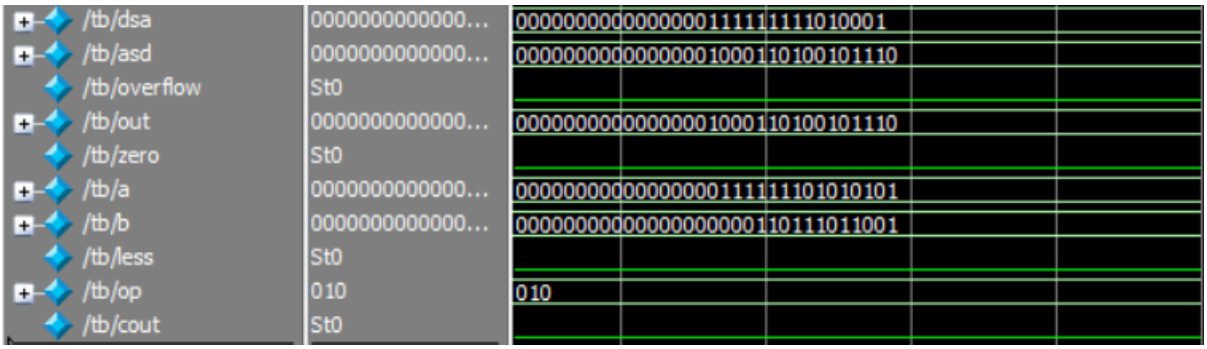


a=00000000000000000111111101010101

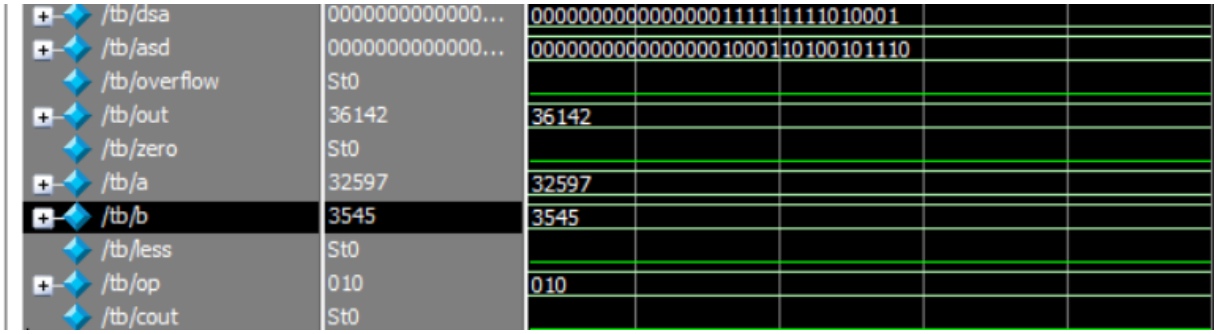
b=00000000000000000000110111011001

op=001 ->or operation

out=00000000000000000111111111011101



BİNARY

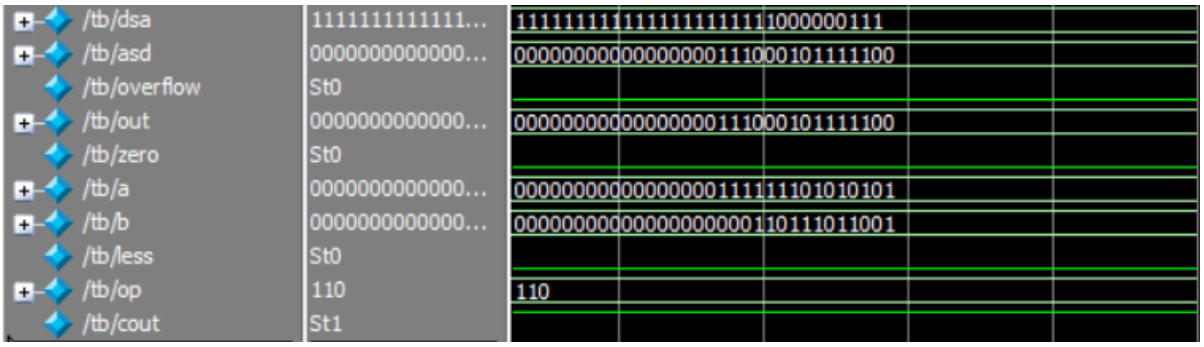


DECİMAL

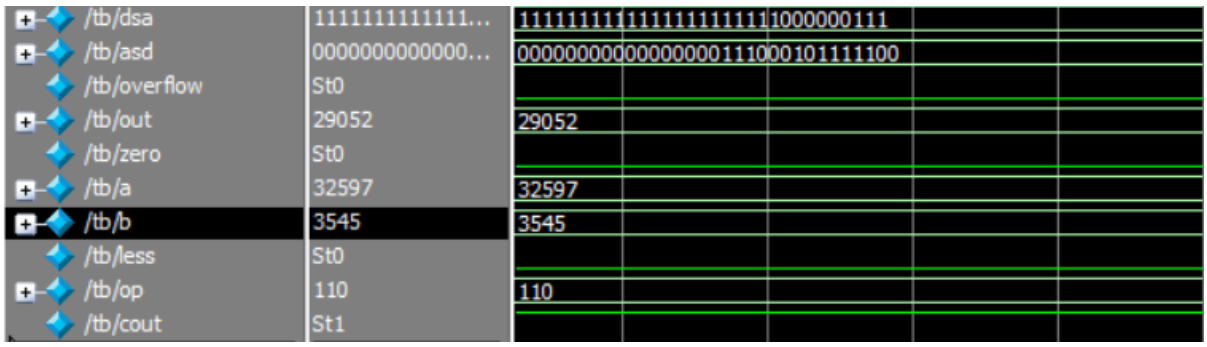
a=00000000000000000111111101010101

b=00000000000000000000110111011001

op=010 ->ADD operation



BİNARY

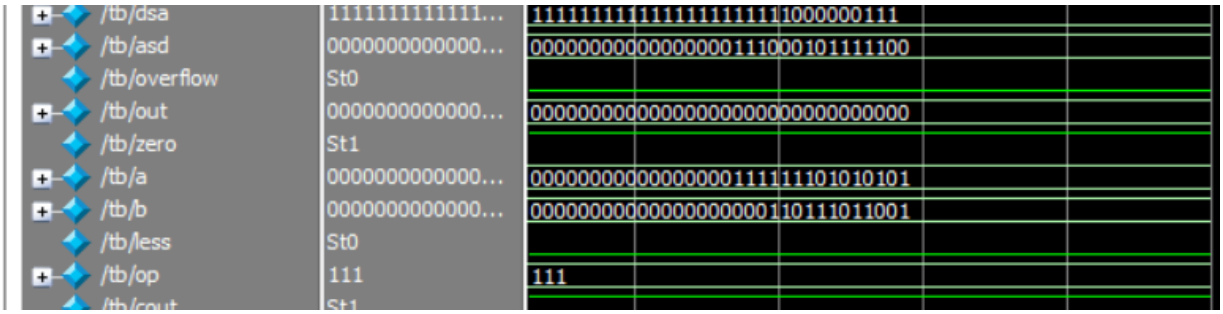


DECİMAL

a=00000000000000000111111101010101

b=00000000000000000000110111011001

op=110 ->SUB operation



a=00000000000000000111111101010101

b=00000000000000000000110111011001

op=111 ->SLT operation

a>b so result =0