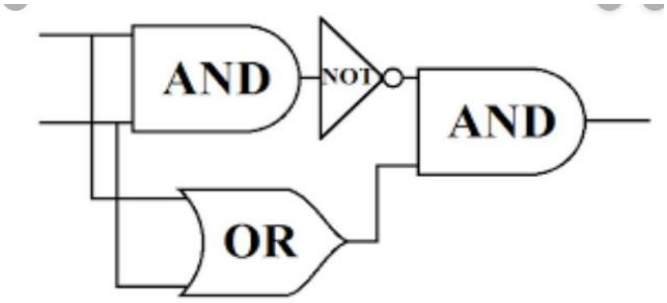
Report HW2

XorGate:

```
module xorGate(output result, input a, input b);
        wire w1,w2,w3;
        and and1(w1, a, b);
        not not1(w2, w1);
        or or1(w3, a, b);
        and and2(result, w2, w3);
endmodule
```

2 tane and , 1 tane not ve 1 tane or gate kullanilmistir. Tasarimim asagidaki design gibidir. Toplam 4 tane gate kullanilmistir.



XorGate Testbench:

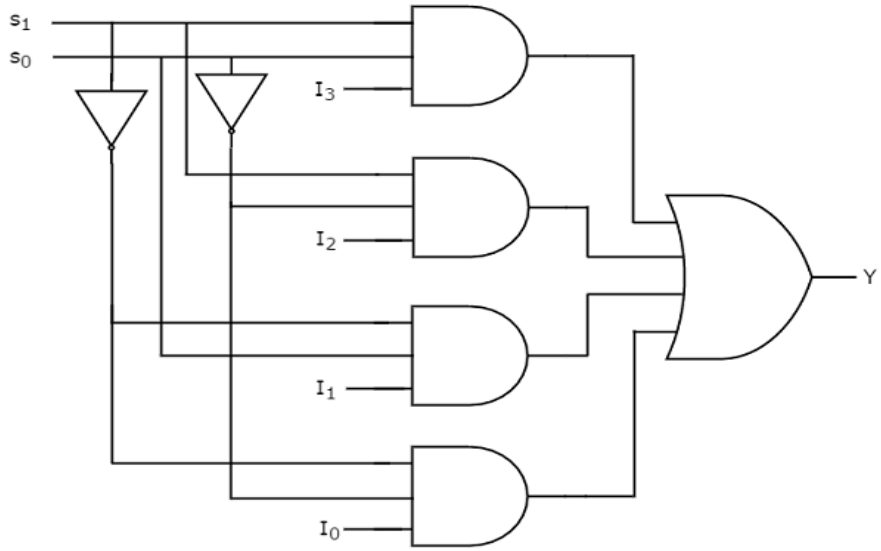
Verilebilecek butun degerleri denenmistir. Elde edilen sonuclar:

```
sim:/testbenchxorGate/result
VSIM 9> step -current
# time =  0, result = 0, a = 0, a = 0
# time = 20, result = 1, a = 0, a = 1
# time = 40, result = 1, a = 1, a = 0
# time = 60, result = 0, a = 1, a = 1
```

Mux 4x1 :

```
module mux_4x1(output result,input I0,input I1,input I2,input I3,input s1,input s0);
        wire nots0, nots1;
        wire w1, w2, w3, w4;
        not n1(nots0, s0);
        not n2(nots1, s1);
        and and1(w1, I0, nots1, nots0);
        and and2(w2, I1, nots1, s0);
        and and3(w3, I2, s1, nots0);
        and and4(w4, I3, s1, s0);
        or or1(result, w1, w2, w3, w4);
endmodule
```

4 tane girisi ve 2 tane select ucu vardir. Select uclarinin not degerini alabilmek icin 2 tane not gate kullanilmistir. 4 tane and ve 1 tane or gate kullanildi. Toplam 7 tane gate kullanilmistir.
Tasarimim asagidaki sekildedir:



Mux 4x1 testbench:

Elde edilen sonuclar:

```
sim:/testbenchMux_4x1/result
VSIM 12> step -current
# time =  0, result = 0, I0 = 0, I1 = 1, I2 = 1, I3 = 0, s1 = 0, s0 = 0
# time = 20, result = 1, I0 = 0, I1 = 1, I2 = 1, I3 = 0, s1 = 0, s0 = 1
# time = 40, result = 1, I0 = 0, I1 = 1, I2 = 1, I3 = 0, s1 = 1, s0 = 0
# time = 60, result = 0, I0 = 0, I1 = 1, I2 = 1, I3 = 0, s1 = 1, s0 = 1
```

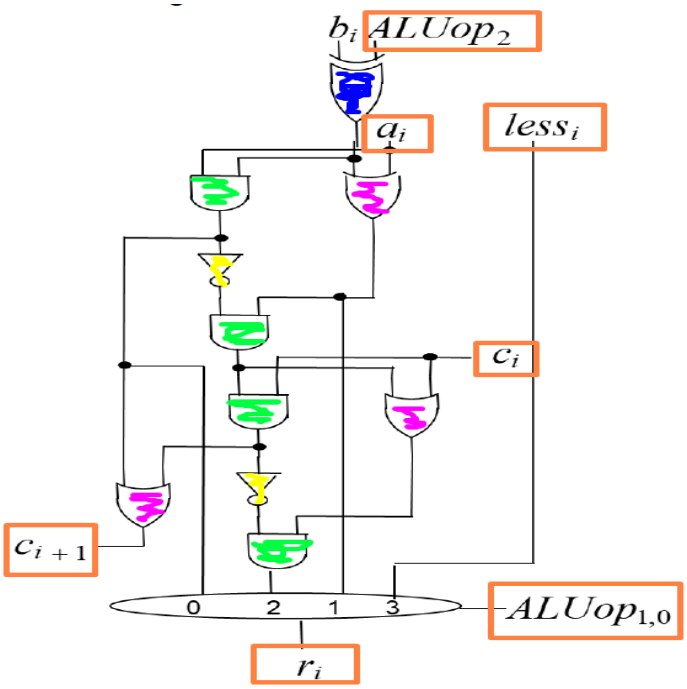
ALU 1 Bit:

```
module ALU1bit(result, carryOut, a, b, opCode, carryIn, lessi);
        input a, b, carryIn, lessi;
        input [2:0] opCode;
        output carryOut, result;
        wire w1, w2, w3, w4, w5, w6, w7, w8, w9;
        xorGate xor1(w1, opCode[2], b);
        or or1(w2, w1, a);
        and and1(w3, w1, a);
        not not1(w4, w3);
        and and2(w5, w4, w2);
        and and3(w6, w5, carryIn);
        or or2(carryOut, w6, w3);
        not not2(w7, w6);
        or or3(w8, w5, carryIn);
        and and4(w9, w8, w7);
        mux_4x1 mux1(result, w3, w2, w9, lessi, opCode[1], opCode[0]);
```

endmodule

Hocanin bize verdigi, pdfdeki tasarim kullanilmistir. 1 tane xor, 3 tane or, 4 tane and, 2 tane not gate kullanildi bu tasarim icin. 1 tane de 4x1'lik mux kullanildi.Toplam gate sayisi 10'dur.

Kullandigim tasarim:



ai (a), bi (b), ci (carryIn), lessi, ALUOp[1:0] ve ALUOp[2] degerleri giris degerleridir. Ci+1 (carryOut) ve ri (result) degerleri output degerleridir.

Function	Binvert (1 line)	Operation (2 lines)
and	0	00
or	0	01
add	0	10
subtract	1	10
slt	1	11

Yan taraftaki gorsel opCode degerlerini gostermektedir. Binvert diye adlandirilan bit ALUOp[2] bitidir. Operation ise ALUOp[1:0] bitleridir.

ALU1Bit Testbench:

```
VSIM 15> step -current
# time =  0, a = 1, b = 1, opCode = 000, carryIn = x, result = 1, carryOut = 1, lessi = x
# time = 20, a = 0, b = 1, opCode = 000, carryIn = x, result = 0, carryOut = x, lessi = x
# time = 40, a = 1, b = 0, opCode = 000, carryIn = x, result = 0, carryOut = x, lessi = x
# time = 60, a = 0, b = 0, opCode = 000, carryIn = x, result = 0, carryOut = 0, lessi = x
# time = 80, a = 1, b = 1, opCode = 001, carryIn = x, result = 1, carryOut = 1, lessi = x
# time = 100, a = 0, b = 1, opCode = 001, carryIn = x, result = 1, carryOut = x, lessi = x
# time = 120, a = 1, b = 0, opCode = 001, carryIn = x, result = 1, carryOut = x, lessi = x
# time = 140, a = 0, b = 0, opCode = 001, carryIn = x, result = 0, carryOut = 0, lessi = x
# time = 160, a = 1, b = 1, opCode = 010, carryIn = 0, result = 0, carryOut = 1, lessi = x
# time = 180, a = 0, b = 1, opCode = 010, carryIn = 0, result = 1, carryOut = 0, lessi = x
# time = 200, a = 1, b = 0, opCode = 010, carryIn = 0, result = 1, carryOut = 0, lessi = x
# time = 220, a = 0, b = 0, opCode = 010, carryIn = 0, result = 0, carryOut = 0, lessi = x
# time = 240, a = 1, b = 1, opCode = 110, carryIn = 1, result = 0, carryOut = 1, lessi = x
# time = 260, a = 0, b = 1, opCode = 110, carryIn = 1, result = 1, carryOut = 0, lessi = x
# time = 280, a = 1, b = 0, opCode = 110, carryIn = 1, result = 1, carryOut = 1, lessi = x
# time = 300, a = 0, b = 0, opCode = 110, carryIn = 1, result = 0, carryOut = 1, lessi = x
# time = 320, a = 1, b = 0, opCode = 111, carryIn = 1, result = 0, carryOut = 1, lessi = x
# time = 340, a = 0, b = 1, opCode = 111, carryIn = 1, result = 0, carryOut = 0, lessi = x

VSIM 16>
```

Lessi degerine parametre girerken 1'b0 degeri verildigi icin x seklinde gozukmektedir.

CarryIn degerleri:

And ve or icin don’t care durumu vardir. Add durumu icin 0 girilmeli, subtraction ise 1 girilmelidir. Set less on than icin de 1 degeri girilmistir.

ALU32bits:

```
module ALU32bits(result, a, b, opCode);

  input [31:0] a, b;
  input [2:0] opCode;
  output [31:0] result;
  wire c[32:1];
  wire overflow, set;
  ALU1bit alu0(result[0], c[1], a[0], b[0], opCode, opCode[2], set);
  ALU1bit alu1(result[1], c[2], a[1], b[1], opCode, c[1], 1'b0);
  ALU1bit alu2(result[2], c[3], a[2], b[2], opCode, c[2], 1'b0);
  ALU1bit alu3(result[3], c[4], a[3], b[3], opCode, c[3], 1'b0);
  ALU1bit alu4(result[4], c[5], a[4], b[4], opCode, c[4], 1'b0);
  ALU1bit alu5(result[5], c[6], a[5], b[5], opCode, c[5], 1'b0);
  ALU1bit alu6(result[6], c[7], a[6], b[6], opCode, c[6], 1'b0);
  ALU1bit alu7(result[7], c[8], a[7], b[7], opCode, c[7], 1'b0);
  ALU1bit alu8(result[8], c[9], a[8], b[8], opCode, c[8], 1'b0);
  ALU1bit alu9(result[9], c[10], a[9], b[9], opCode, c[9], 1'b0);
  ALU1bit alu10(result[10], c[11], a[10], b[10], opCode, c[10], 1'b0);
  ALU1bit alu11(result[11], c[12], a[11], b[11], opCode, c[11], 1'b0);
  ALU1bit alu12(result[12], c[13], a[12], b[12], opCode, c[12], 1'b0);
  ALU1bit alu13(result[13], c[14], a[13], b[13], opCode, c[13], 1'b0);
  ALU1bit alu14(result[14], c[15], a[14], b[14], opCode, c[14], 1'b0);
  ALU1bit alu15(result[15], c[16], a[15], b[15], opCode, c[15], 1'b0);
  ALU1bit alu16(result[16], c[17], a[16], b[16], opCode, c[16], 1'b0);
  ALU1bit alu17(result[17], c[18], a[17], b[17], opCode, c[17], 1'b0);
  ALU1bit alu18(result[18], c[19], a[18], b[18], opCode, c[18], 1'b0);
  ALU1bit alu19(result[19], c[20], a[19], b[19], opCode, c[19], 1'b0);
  ALU1bit alu20(result[20], c[21], a[20], b[20], opCode, c[20], 1'b0);
  ALU1bit alu21(result[21], c[22], a[21], b[21], opCode, c[21], 1'b0);
  ALU1bit alu22(result[22], c[23], a[22], b[22], opCode, c[22], 1'b0);
  ALU1bit alu23(result[23], c[24], a[23], b[23], opCode, c[23], 1'b0);
  ALU1bit alu24(result[24], c[25], a[24], b[24], opCode, c[24], 1'b0);
  ALU1bit alu25(result[25], c[26], a[25], b[25], opCode, c[25], 1'b0);
  ALU1bit alu26(result[26], c[27], a[26], b[26], opCode, c[26], 1'b0);
  ALU1bit alu27(result[27], c[28], a[27], b[27], opCode, c[27], 1'b0);
  ALU1bit alu28(result[28], c[29], a[28], b[28], opCode, c[28], 1'b0);
  ALU1bit alu29(result[29], c[30], a[29], b[29], opCode, c[29], 1'b0);
  ALU1bit alu30(result[30], c[31], a[30], b[30], opCode, c[30], 1'b0);
  msb1bit msb1(result[31], c[32], a[31], b[31], opCode, c[31], 1'b0, overflow, set);
endmodule
```

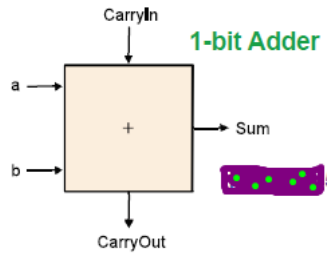
32 tane ALU1bit’lik design kullanilmistir. Ve en son kisimda ise most significant biti bulabilmek icin ayri bir design yapilmistir.

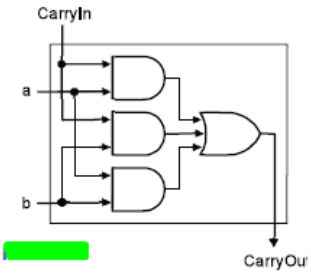
Msb1bit (Most Significant Bit) :

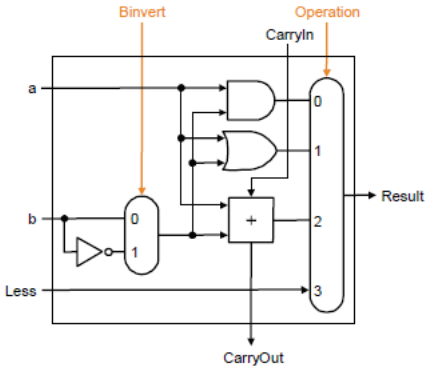
```
module msb1bit(result, carryOut, a, b, opCode, carryIn, lessi, overflow, set);
        input a, b, carryIn, lessi;
        input [2:0] opCode;
        output result, carryOut, overflow, set;
        wire w1,w2,w3,w4,w5,w6,w7,w8,w9;
        xorGate xor1(w1, opCode[2], b);
        or or1(w2, w1, a);
        and and1(w3, w1, a);
        not not1(w4, w3);
        and and2(w5, w4, w2);
        and and3(w6, w5, carryIn);
        or or2(carryOut, w6, w3);
        not not2(w7, w6);
        or or3(w8, w5, carryIn);
        and and4(w9, w8, w7);
        mux_4x1 mux1(result, w3, w2, w9, lessi, opCode[1], opCode[0]);
        xorGate xor2(overflow, carryIn, carryOut);
        xorGate xor3(set, overflow, w9);
endmodule
```

Bu tasarim set less on than isleminin gerceklesebilmesi icin gereklidir. ALU1bit kisminin neredeyse aynisidir sadece en alt kisma 2 tane xor eklenip gerekli islemler gerceklestirilmistir. Set degerine ihtiyacimiz vardir bunun icin ise overflow degeri bulunmalidir. Xor gateleri bunun icin kullanilmistir. Buldugumuz set degeri ilk ALU1bit lessi degerine atanmistir. (Orada kullanilmistir.)

Tasarim kismi:

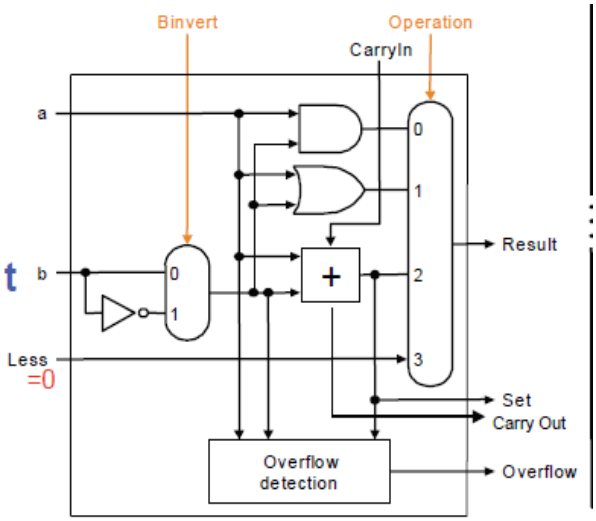


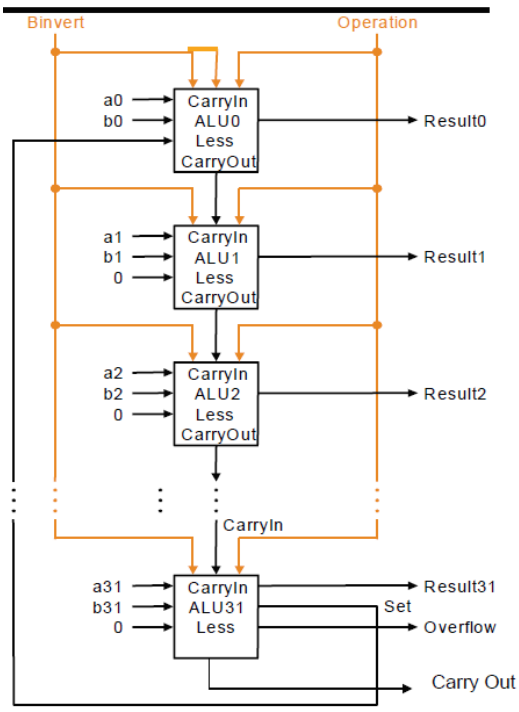




Bu tasarimlar ALU1bit tasarimidir. (Bizim tasarimizla aynidir.)

Simdi ise gosterilecek tasarim neden most significant bite ihtiyacimiz oldugunun gosterimidir:





Tasarimdan da anlasilacagi uzere, most significant
degerinin ALU1bit designindaki output olan set degeri, input olarak kullanilmalidir. Bunu yapmamizin nedeni set less than instructionunu kullabilmek icindir. Diger lessi degerleri 0 iken less0'a set degeri atanmalidir.

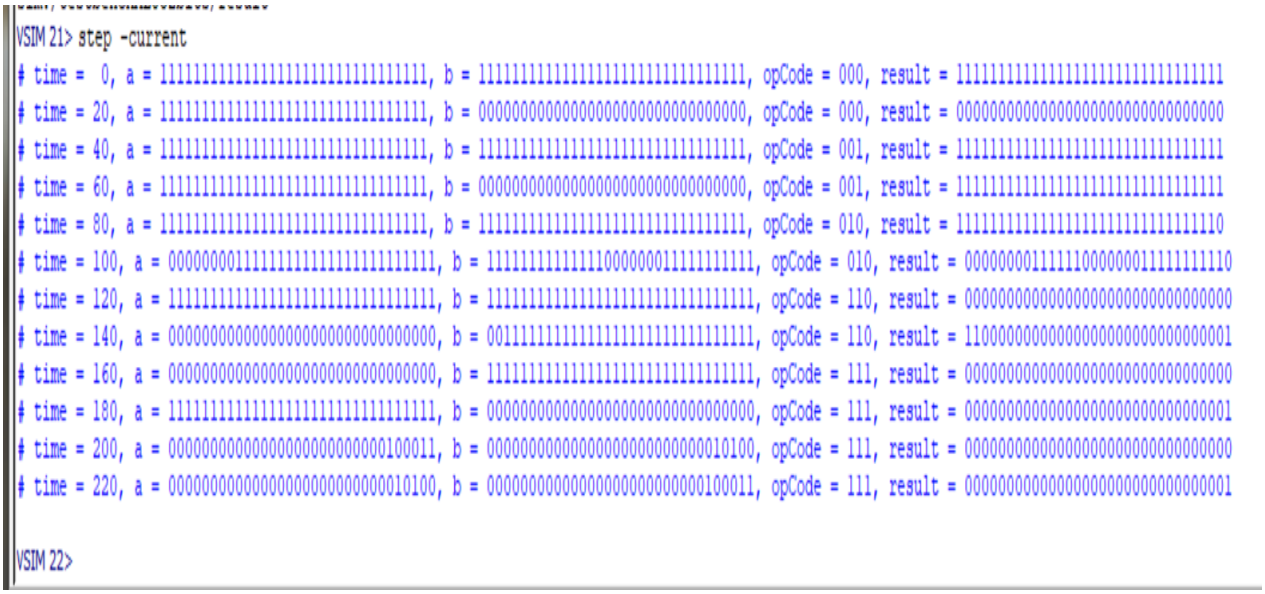
Msb1bit Testbench:

```
VSIM 18> step -current
# time =  0, a = 1, b = 1, opCode = 000, carryIn = 0, overflow = 1, set = 1, carryOut = 1, result = 1
# time = 20, a = 1, b = 1, opCode = 000, carryIn = 1, overflow = 0, set = 1, carryOut = 1, result = 1
# time = 40, a = 0, b = 0, opCode = 000, carryIn = 0, overflow = 0, set = 0, carryOut = 0, result = 0
# time = 60, a = 1, b = 0, opCode = 010, carryIn = 0, overflow = 0, set = 1, carryOut = 0, result = 1
# time = 80, a = 0, b = 1, opCode = 010, carryIn = 0, overflow = 0, set = 1, carryOut = 0, result = 1
# time = 100, a = 1, b = 1, opCode = 010, carryIn = 1, overflow = 0, set = 1, carryOut = 1, result = 1
# time = 120, a = 1, b = 1, opCode = 111, carryIn = 1, overflow = 0, set = 0, carryOut = 1, result = 0
# time = 140, a = 0, b = 1, opCode = 111, carryIn = 1, overflow = 1, set = 0, carryOut = 0, result = 0
# time = 160, a = 1, b = 1, opCode = 111, carryIn = 0, overflow = 0, set = 1, carryOut = 0, result = 0
# time = 180, a = 0, b = 1, opCode = 111, carryIn = 0, overflow = 0, set = 0, carryOut = 0, result = 0

VSIM 19>
```

Yazdigim test degerlerine gore aldigim sonuclardir.

ALU32bits Testbench:



Butun opCode degerlerini, degisik 32 bitlik degerler icin denenmistir. Denedigim degerlere gore aldigim sonuclardir.