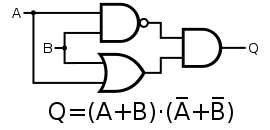
# Computer organization : CSE – 331

# #ASSİGNMENT 2 – REPORT

*AUTHOR* : *Lemye Ceren Gümüş – 151044071*

1. First xor module is created.



module XOR\_module(res,input1,input2);

input input1,input2;

output res;

wire n1,n2,res1,res2;

not not1(n1,input1);

not not2(n2,input2);

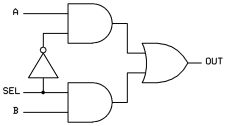
and a1(res1,n1,input2);

and a2(res2,n2,input1);

or o1(res,res1,res2);

endmodule

1. After mux2 is designed



module MUX\_2(result,input1,input2,select);

input input1,input2,select;

output result;

wire select\_not, for\_select\_input1, for\_select\_input2;

not s1(select\_not,select);

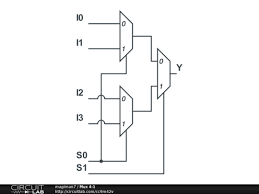
and o2(for\_select\_input2,select,input2);

and o1(for\_select\_input1,select\_not,input1);

or res(result,for\_select\_input1,for\_select\_input2);

endmodule

1. After mux4 is designed



module MUX\_4(result, select, input0,input1,input2,input3 );

input input0,input1,input2,input3;

input [1:0] select;

output result;

wire for\_mux1, for\_mux2;

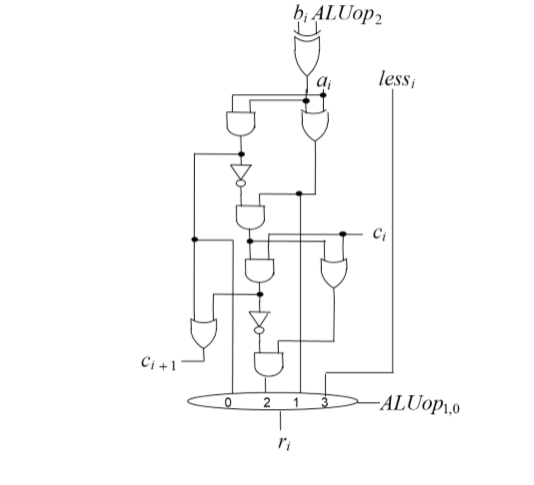
MUX\_2 r1(for\_mux1,input0,input1,select[0]);

MUX\_2 r2(for\_mux2,input2,input3,select[0]);

MUX\_2 res(result,for\_mux1,for\_mux2,select[1]);

Endmodule

1. 1 bit alu module is written



module ALU\_1(res,aluop,a,b,carryin,carryout,less);

input a,b,carryin;

input [2:0] aluop;

output res,carryout,less;

wire r1,r2,r3,r4,v;

XOR\_module x1(r1,b,aluop[2]);

/\*\*\*\*3 r2\*\*///

or o1(r2,r1,a);

and a1(r3,r1,a);

not n1(r4,r3);

and a2(r5,r4,r2);

and a3(r6,r5,carryin);

or o2(r7,r5,carryin);

not n2(r8,r6);

/\*\*\*\*\*2\*//////

and a4(r9,r8,r7);

/\*\*\*\*0\* r3\*///

or o3(carryout,r3,r6);

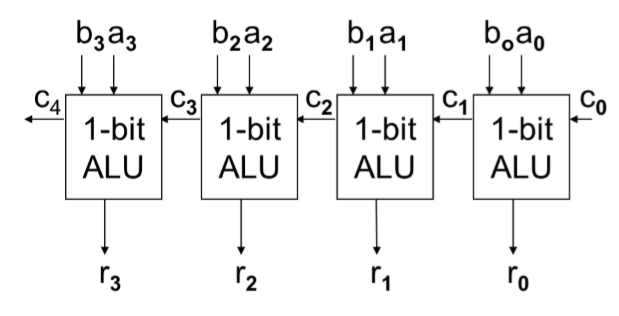
XOR\_module X3(v,carryin,carryout);

XOR\_module X4(less,r9,v);

MUX\_4 m4(res,aluop[1:0],r3,r2,r9,less);

endmodule

1. After that alu\_32 is defined



module ALU\_32(res,a,b,c,aluop,less,carryout);

input [0:31] a;

input [0:31] b;

input [0:2] aluop;

input c;

output [0:31] res;

output carryout,less;

wire [0:31] res1;

wire [0:31] less1;

wire c0,c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,c12,c13,c14,c15,c16,c17,c18,c19,c20,c21,c22,c23,c24,c25,c26,c27,c28,c29,c30;

ALU\_1 a1(res1[31],aluop,a[31],b[31],c,c0,less);

MUX\_8 M1(res[31],aluop,res1[31],res1[31],res1[31],res1[31],res1[31],res1[31],res1[31],less);

ALU\_1 a2(res1[30],aluop,a[30],b[30],c0,c1,less1[31]);

MUX\_8 M2(res[30],aluop,res1[30],res1[30],res1[30],res1[30],res1[30],res1[30],res1[30],0);

ALU\_1 a3(res1[29],aluop,a[29],b[29],c1,c2,less1[30]);

MUX\_8 M3(res[29],aluop,res1[29],res1[29],res1[29],res1[29],res1[29],res1[29],res1[29],0);

ALU\_1 a4(res1[28],aluop,a[28],b[28],c2,c3,less1[29]);

MUX\_8 M4(res[28],aluop,res1[28],res1[28],res1[28],res1[28],res1[28],res1[28],res1[28],0);

ALU\_1 a5(res1[27],aluop,a[27],b[27],c3,c4,less1[28]);

MUX\_8 M5(res[27],aluop,res1[27],res1[27],res1[27],res1[27],res1[27],res1[27],res1[27],0);

ALU\_1 a6(res1[26],aluop,a[26],b[26],c4,c5,less1[27]);

MUX\_8 M6(res[26],aluop,res1[26],res1[26],res1[26],res1[26],res1[26],res1[26],res1[26],0);

ALU\_1 a7(res1[25],aluop,a[25],b[25],c5,c6,less1[26]);

MUX\_8 M6h(res[25],aluop,res1[25],res1[25],res1[25],res1[25],res1[25],res1[25],res1[25],0);

ALU\_1 a8(res1[24],aluop,a[24],b[24],c6,c7,less1[25]);

MUX\_8 M7(res[24],aluop,res1[24],res1[24],res1[24],res1[24],res1[24],res1[24],res1[24],0);

ALU\_1 a9(res1[23],aluop,a[23],b[23],c7,c8,less1[24]);

MUX\_8 M8(res[23],aluop,res1[23],res1[23],res1[23],res1[23],res1[23],res1[23],res1[23],0);

ALU\_1 a10(res1[22],aluop,a[22],b[22],c8,c9,less1[23]);

MUX\_8 M9(res[22],aluop,res1[22],res1[22],res1[22],res1[22],res1[22],res1[22],res1[25],0);

ALU\_1 a11(res1[21],aluop,a[21],b[21],c9,c10,less1[22]);

MUX\_8 M10(res[21],aluop,res1[21],res1[21],res1[21],res1[21],res1[21],res1[21],res1[21],0);

ALU\_1 a12(res1[20],aluop,a[20],b[20],c10,c11,less1[21]);

MUX\_8 M12(res[20],aluop,res1[20],res1[20],res1[20],res1[20],res1[20],res1[20],res1[20],0);

ALU\_1 a13(res1[19],aluop,a[19],b[19],c11,c12,less1[20]);

MUX\_8 M1x(res[19],aluop,res1[19],res1[19],res1[19],res1[19],res1[19],res1[19],res1[19],0);

ALU\_1 a14(res1[18],aluop,a[18],b[18],c12,c13,less1[19]);

MUX\_8 M14(res[18],aluop,res1[18],res1[18],res1[18],res1[18],res1[18],res1[18],res1[18],0);

ALU\_1 a15(res1[17],aluop,a[17],b[17],c13,c14,less1[18]);

MUX\_8 M15(res[17],aluop,res1[17],res1[17],res1[17],res1[17],res1[17],res1[17],res1[17],0);

ALU\_1 a16(res1[16],aluop,a[16],b[16],c14,c15,less1[17]);

MUX\_8 M16(res[16],aluop,res1[16],res1[16],res1[16],res1[16],res1[16],res1[16],res1[16],0);

ALU\_1 a17(res1[15],aluop,a[15],b[15],c15,c16,less1[16]);

MUX\_8 M17(res[15],aluop,res1[15],res1[15],res1[15],res1[15],res1[15],res1[15],res1[15],0);

ALU\_1 a18(res1[14],aluop,a[14],b[14],c16,c17,less1[15]);

MUX\_8 M18(res[14],aluop,res1[14],res1[14],res1[14],res1[14],res1[14],res1[14],res1[14],0);

ALU\_1 a19(res1[13],aluop,a[13],b[13],c17,c18,less1[14]);

MUX\_8 M19(res[13],aluop,res1[13],res1[13],res1[13],res1[13],res1[13],res1[13],res1[13],0);

ALU\_1 a20(res1[12],aluop,a[12],b[12],c18,c19,less1[13]);

MUX\_8 M20(res[12],aluop,res1[12],res1[12],res1[12],res1[12],res1[12],res1[12],res1[12],0);

ALU\_1 a21(res1[11],aluop,a[11],b[11],c19,c20,less1[12]);

MUX\_8 M21(res[11],aluop,res1[11],res1[11],res1[11],res1[11],res1[11],res1[11],res1[11],0);

ALU\_1 a22(res1[10],aluop,a[10],b[10],c20,c21,less1[11]);

MUX\_8 M22(res[10],aluop,res1[10],res1[10],res1[10],res1[10],res1[10],res1[10],res1[10],0);

ALU\_1 a23(res1[9],aluop,a[9],b[9],c21,c22,less1[10]);

MUX\_8 M23(res[9],aluop,res1[9],res1[9],res1[9],res1[9],res1[9],res1[9],res1[9],0);

ALU\_1 a24(res1[8],aluop,a[8],b[8],c22,c23,less1[9]);

MUX\_8 M24(res[8],aluop,res1[8],res1[8],res1[8],res1[8],res1[8],res1[8],res1[8],0);

ALU\_1 a25(res1[7],aluop,a[7],b[7],c23,c24,less1[8]);

MUX\_8 M25(res[7],aluop,res1[7],res1[7],res1[7],res1[7],res1[7],res1[7],res1[7],0);

ALU\_1 a26(res1[6],aluop,a[6],b[6],c24,c25,less1[7]);

MUX\_8 M26(res[6],aluop,res1[6],res1[6],res1[6],res1[6],res1[6],res1[6],res1[6],0);

ALU\_1 a27(res1[5],aluop,a[5],b[5],c25,c26,less1[6]);

MUX\_8 M27(res[5],aluop,res1[5],res1[5],res1[5],res1[5],res1[5],res1[5],res1[5],0);

ALU\_1 a28(res1[4],aluop,a[4],b[4],c26,c27,less1[5]);

MUX\_8 M28(res[4],aluop,res1[4],res1[4],res1[4],res1[4],res1[4],res1[4],res1[4],0);

ALU\_1 a29(res1[3],aluop,a[3],b[3],c27,c28,less1[4]);

MUX\_8 M29(res[3],aluop,res1[3],res1[3],res1[3],res1[3],res1[3],res1[3],res1[3],0);

ALU\_1 a30(res1[2],aluop,a[2],b[2],c28,c29,less1[3]);

MUX\_8 M30(res[2],aluop,res1[2],res1[2],res1[2],res1[2],res1[2],res1[2],res1[2],0);

ALU\_1 a31(res1[1],aluop,a[1],b[1],c29,c30,less1[2]);

MUX\_8 M31(res[1],aluop,res1[1],res1[1],res1[1],res1[1],res1[1],res1[1],res1[1],0);

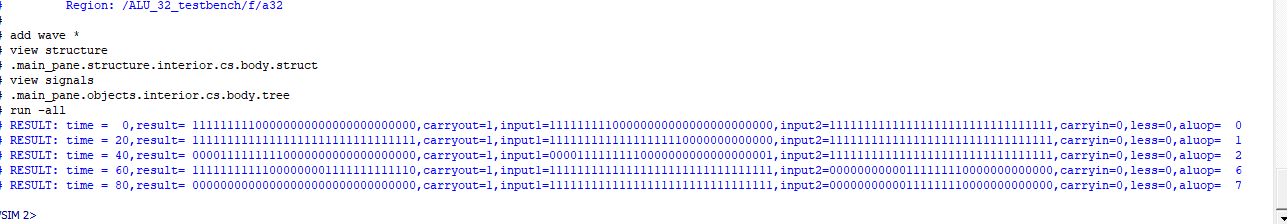
ALU\_1 a32(res1[0],aluop,a[0],b[0],c30,carryout,less1[1]);

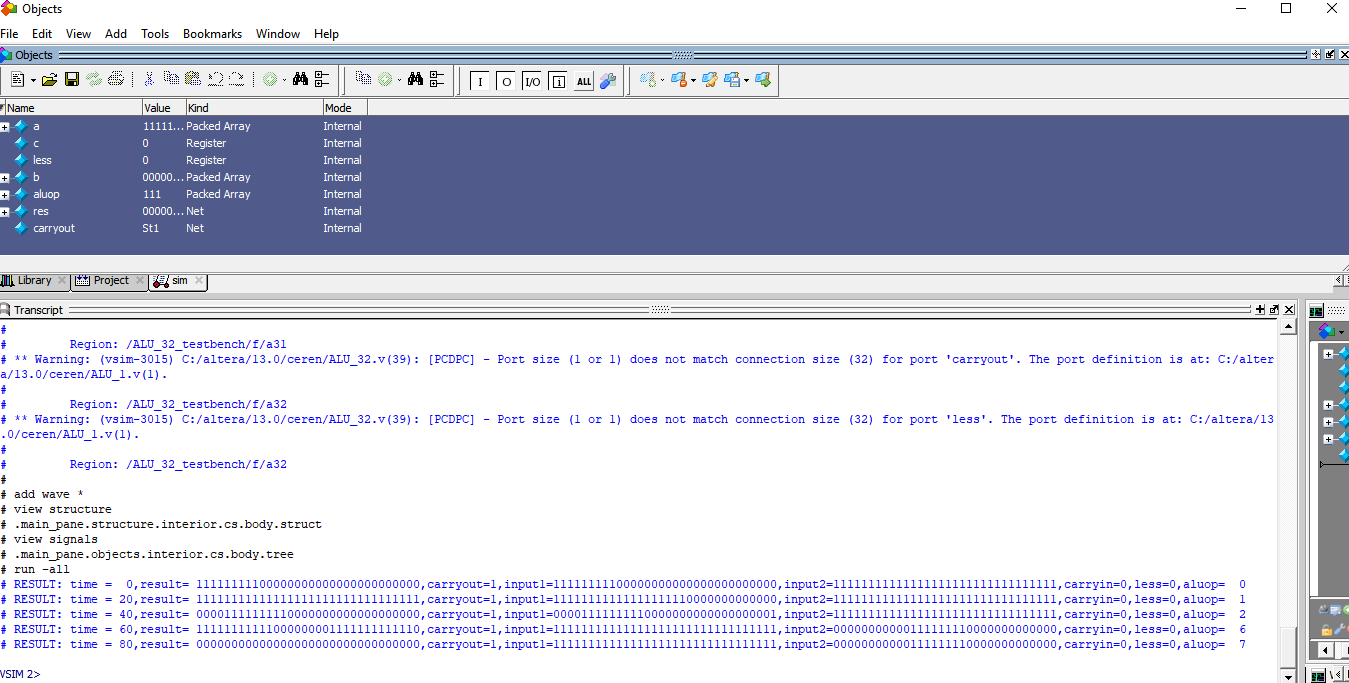
MUX\_8 M32(res[0],aluop,res1[0],res1[0],res1[0],res1[0],res1[0],res1[0],res1[0],0);

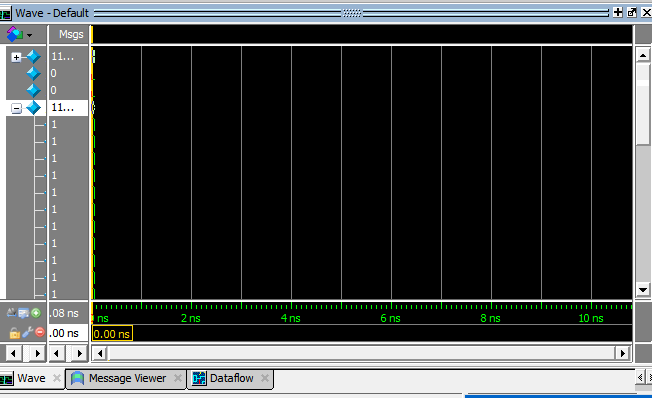
endmodule

# Module output :

Alu \_32\_module :







**FGPA Board Demo**

Pin Planner ‘da set edilen pinler aşağıda gösterilmiştir. Toplam 10 tane toggle switch butonu olduğu için 3 bitlik ALU üzerinde demo oluşturulmuştur . ALU\_3\_fpga\_demo.v dosyasının içerisinde ALU\_3 çağrılmıştır 3 bitlik A (D2,E4,E3) ve 3 bitlik B için (H7,J7,G5) pinleri set edilmiştir Sonuçlar led üzerinde J1,J2,J3 pinlerine set edilmiştir.

module ALU\_3\_fpga\_demo

(

// {ALTERA\_ARGS\_BEGIN} DO NOT REMOVE THIS LINE!

a,

aluop,

b,

c,

carryout,

less,

res

// {ALTERA\_ARGS\_END} DO NOT REMOVE THIS LINE!

);

// {ALTERA\_IO\_BEGIN} DO NOT REMOVE THIS LINE!

input [0:2] a;

input [0:2] aluop;

input [0:2] b;

input c;

output carryout;

output less;

output [0:2] res;

ALU\_3 test(a,aluop,b,c,carryout,less,res);

// {ALTERA\_IO\_END} DO NOT REMOVE THIS LINE!

// {ALTERA\_MODULE\_BEGIN} DO NOT REMOVE THIS LINE!

// {ALTERA\_MODULE\_END} DO NOT REMOVE THIS LINE!

endmodule

