Colt Kiser & Nathan St. Amour

EE 3613

Homework 4

1.)

```
module ALU16Bit(a, b, cin, less, op, result, cout, set, zero, g, p, overflow);
input [15:0] a, b;
input cin, less;
input [2:0] op;                    // op[2] is "binv". op[1:0] denotes the 2-bit operation.
output [15:0] result;
output cout, set, zero, g, p, overflow;
// set is the result of the most-significant
// ADDER unit. zero is 1 if the result is 0x0000.
// Otherwise, it is 0.
wire [3:0] c;       // cout from each 4-bit ALU
wire g0, g1, g2,g3, p0, p1, p2, p3; // G and P from each 4-bit ALU
wire [3:0] setv;    // set from each 4-bit ALU
wire [3:0] ovf; // overflow from each 4-bit ALU
wire [3:0] z;       // zero from each 4-bit ALU
wire C1, C2, C3, C4;


//module    FourBitALU(       a,       b, cin,    less, op,       result,  cout, G, P,     set, overflow, zero);
FourBitALU FourBitALU0(  a[3:0], b[3:0],    cin,    less, op,  result[3:0], c[0], g0, p0, setv[0], ovf[0], z[0]);
FourBitALU FourBitALU1(  a[7:4],  b[7:4],  C1,    1'b0, op,  result[7:4], c[1], g1, p1, setv[1], ovf[1], z[1]);
FourBitALU FourBitALU2( a[11:8], b[11:8],  C2,    1'b0, op, result[11:8], c[2], g2, p2, setv[2], ovf[2], z[2]);
FourBitALU FourBitALU3(a[15:12],b[15:12],  C3,    1'b0, op, result[15:12], c[3], g3, p3, setv[3], ovf[3], z[3]);

//module CLA(g0, p0, g1, p1, g2, p2, g3, p3, cin, C1, C2, C3, C4, G, P);
CLA CLA0(g0, p0, g1, p1, g2, p2, g3, p3, cin, C1, C2, C3, C4, g, p);

assign cout=C4;
assign set=setv[3];
assign zero=z[0]&z[1]&z[2]&z[3];
assign overflow=c[3];

endmodule
```

Test Bench

```
`timescale 1ns / 1ps
module ALU16Bitb();
    reg [15:0] a, b;
    reg cin, less;
    reg [2:0] op;
    wire [15:0] result;
    wire cout, set, zero, g, p, overflow;

    ALU16Bit  alu(a, b, cin, less, op, result, cout, set, zero, g, p, overflow);

    initial
        begin
    //Overflow!
    a <= 16'b 1111111111111111; b <= 16'b 1111111111111111; cin <= 0; less <= 0; op <= 3'b010;
    #1 $display ( "Addition:  a = %d, b = %d, cin = %d, op = %d, result = %d, cout = %d, set = %d, zero = %d, g = %d, p = %b, overflow = %d", a, b, cin, op, result, cout, set, zero, g, p, overflow);

    a <= 16'b 0000000001000000; b <= 16'b 0000000000000001; cin <= 0; less <= 0; op <= 3'b010;
    #1 $display ("Addition: a = %d, b = %d, cin = %d, op = %d, result = %d, cout = %d, set = %d, zero = %d, g = %d, p = %b, overflow = %d", a, b, cin, op, result, cout, set, zero, g, p, overflow);

    a <= 16'b 0101010101010101; b <= 16'b 1010101010101010; cin <= 0; less <= 0; op <= 3'b010;
    #1 $display ( "Addition 3 a = %d, b = %d, cin = %d, op = %d, result = %d, cout = %d, set = %d, zero = %d, g = %d, p = %b, overflow = %d", a, b, cin, op, result, cout, set, zero, g, p, overflow);

    a <= 16'b 100000100000001; b <= 16'b 1101001000000101;  cin <= 0; less <= 0; op <= 3'b000;
    #1 $display ( "And: a = %b, b = %b, cin = %d, op = %d, result = %b, cout = %d, set = %d, zero = %d, g = %d, p = %b, overflow = %d", a, b, cin, less, op, result, cout, set, zero, g, p, overflow);


    a <= 16'b 000000000000000; b <= 16'b 0000000000000000;  cin <= 0; less <= 0; op <= 3'b000;
    #1 $display ( "And: a = %b, b = %b, cin = %d, result = %b, cout = %d, set = %d, zero = %d, g = %d, p = %b, overflow = %d", a, b, cin,  result, cout, set, zero, g, p, overflow);



    a <= 16'b 1100000000000011; b <= 16'b 0000000000000000; cin <= 0; less <= 0; op <= 3'b001;
    #1 $display ( "Or a = %b, b = %b, cin = %d, result = %b, cout = %d, set = %d, zero = %d, g = %d, p = %b, overflow = %d", a, b, cin, result, cout, set, zero, g, p, overflow);
        /*  //case : return less     */
```

Output

```
Addition:  a = 65535, b = 65535, cin = 0, op = 2, result = 65535, cout = 1, set = z, zero = 0, g = 1, p = &b, overflow = 11
Addition: a =    64, b =     1, cin = 0, op = 2, result =    65, cout = 0, set = z, zero = 0, g = 0, p = &b, overflow = 00
Addition 3 a = 21845, b = 43690, cin = 0, op = 2, result = 65535, cout = 0, set = z, zero = 0, g = 0, p = &b, overflow = 10
And: a = 0100000100000001, b = 1101001000000101, cin = 0, op = 0, result = 000, cout = 16385, set = 1, zero = z, g = 0, p = &b, overflow = 101
And: a = 0000000000000000, b = 0000000000000000, cin = 0, result = 0000000000000000, cout = 0, set = z, zero = 1, g = 0, p = &b, overflow = 00
Or a = 1100000000000011, b = 0000000000000000, cin = 0, result = 1100000000000011, cout = 0, set = z, zero = 0, g = 0, p = &b, overflow = 00
```

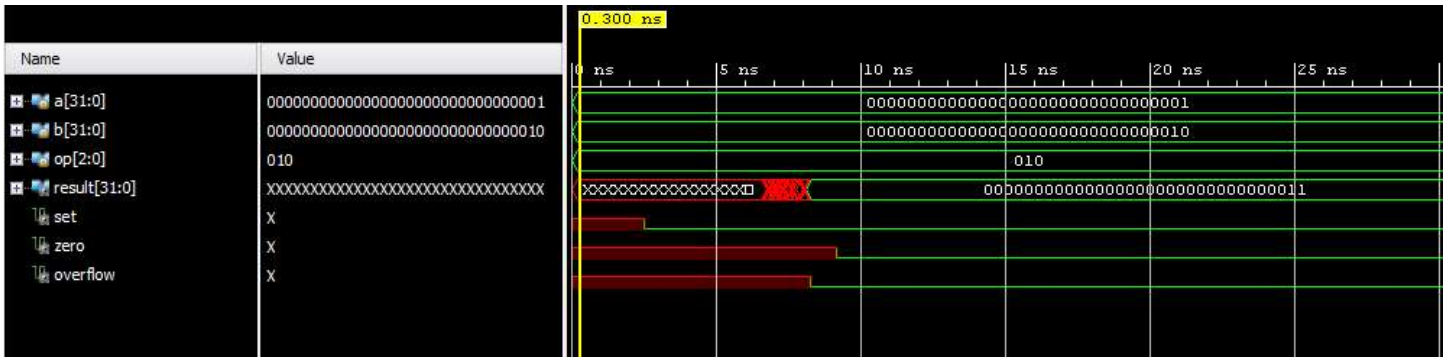
2.)

```
wire s0,s1, over0,over1;
wire [1:0] z;
wire cout0, cout1;
wire zero0, zero1;
//module ALU16Bit(   a,        b,  *cin*, *less*, op,     result*cout*, *set*, zero,      g,     p, overflow);
ALU16Bit alu0( a[15:0],  b[15:0], op[2], s1 , op, result[15:0], cout0, s0 , zero0, g[3:0], p[3:0],  over0);
ALU16Bit alu1(a[31:16], b[31:16],  cout0,  1'b0, op,result[31:16], cout1,  s1, zero1, g[7:4], p[7:4],  over1);

assign overflow = cout1;
assign set = s1;
assign zero = zero0 & zero1;
endmodule
```

Test Bench

```
module thiry_two_tb();
    reg [31:0] a, b;
    reg [2:0] op;
    wire [31:0] result;
    wire set, zero, overflow;

    ALU32Bit my32alu(a, b, op, result, set, zero, overflow);

    initial
        begin
    a <= 32'd 1; b <= 32'd 2;
    op <= 3'b010;
    #1 $display ("a = %d, b = %d, op = %d, result = %d, set = %d, zero = %d, overflow = %d", a, b, op, result, set, zero, overflow);
end

endmodule
```

Waveform



3.)

```
module SignExtension(a, result);
input [15:0] a;             // 16-bit input
output [31:0] result;   // 32-bit output
reg [31:0] result;

always@(a)
  begin
  if(a[15]==1'b1)
    begin
    result={16'b1111111111111111,a};
    end
  else
      begin
      result={16'b0000000000000000,a};
      end
  end

endmodule
```

Test Bench

```
module SignExtension_tb();
    reg [15:0] a;
    wire [31:0] result;

    SignExtension ext(a, result);
    initial
        begin
        a <= 16'd 0;
        #1 $display ( "a = %b, result = %b",  a, result);

        a <= 16'd 23;
        #1 $display ( "a = %b, result = %b",  a, result);
        a <= 16'b 1111111111111111;
        #1 $display ( "a = %b, result = %b",  a, result);
    end
endmodule
```

Output

```
a = 0000000000000000, result = 00000000000000000000000000000000
a = 0000000000010111, result = 00000000000000000000000000010111
a = 1111111111111111, result = 11111111111111111111111111111111
```

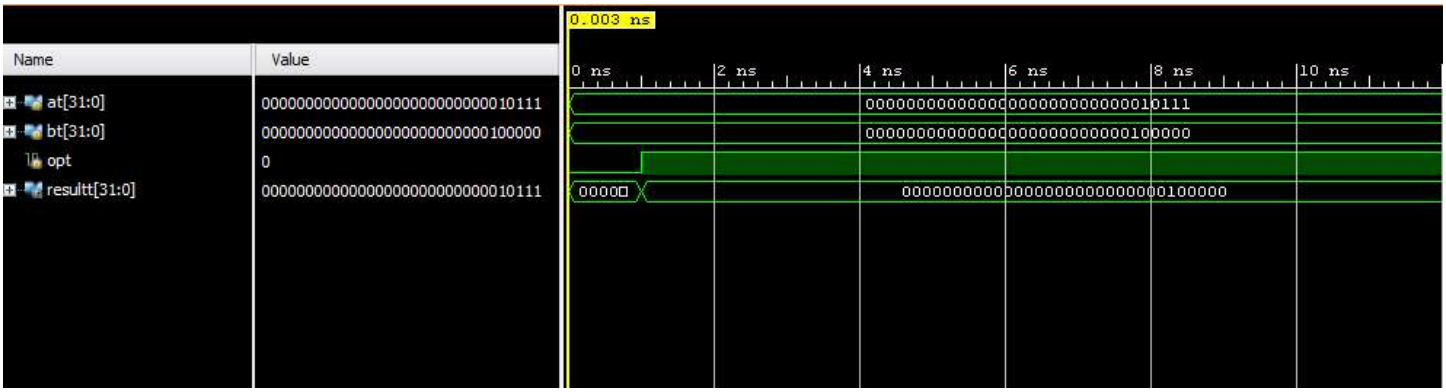
4.)

```
module Mux32Bit2To1(a, b, op, result);
input [31:0] a, b;   // 32-bit inputs
input op;                // one-bit selection i
output [31:0] result; // 32-bit output
reg [31:0]result;
    wire [31:0] a;
    wire [31:0] b;
    wire op;

always@(a,b,op)
begin
if(op==1'b0)
  begin
   result=a;
  end
 else
  begin
   result=b;
 end
 end

endmodule
```

Test Bench

```
module mux32bit_tb();
reg [31:0] at, bt; // 32-bit inputs
    reg opt; // one-bit selection input
    wire [31:0] resultt; // 32-bit output

    Mux32Bit2To1 mymux(at, bt, opt, resultt);

initial begin

    at <= 32'd 23; bt <= 32'd 32 ; opt <= 0;
    #1 $display ( " a = %d, b = %d, op = %b, result = %d", at, bt, opt, resultt);

    at <= 32'd 23; bt <= 32'd 32; opt <= 1;
    #1 $display ( " a = %d, b = %d, op = %b result = %d", at, bt, opt, resultt);

end



endmodule
```

Waveform



Output

```
a =         23, b =         32, op = 0, result =         23
a =         23, b =         32, op = 1 result =         32
```