**ALU.HDL**

CHIP ALU {

IN

x[16], y[16], // 16-bit inputs

zx, // zero the x input?

nx, // negate the x input?

zy, // zero the y input?

ny, // negate the y input?

f, // compute out = x + y (if 1) or x & y (if 0)

no; // negate the out output?

OUT

out[16], // 16-bit output

zr, // 1 if (out == 0), 0 otherwise

ng; // 1 if (out < 0), 0 otherwise

PARTS:

Mux16(a=x,b=false,sel=zx,out=tempx1);//zx

Not16(in=tempx1,out=notx);

Mux16(a=tempx1,b=notx,sel=nx,out=tempx2);//nx

Mux16(a=y,b=false,sel=zy,out=tempy1);//zy

Not16(in=tempy1,out=noty);

Mux16(a=tempy1,b=noty,sel=ny, out=tempy2);//ny

And16(a=tempx2,b=tempy2,out=xAndy);//f=0

Add16(a=tempx2,b=tempy2,out=addxy);//f=1

Mux16(a=xAndy,b=addxy,sel=f,out=tempout1);//f

Not16(in=tempout1,out=notout);

Mux16(a=tempout1,b=notout,sel=no,out=out,out[0..7]=firsthalf, out[8..15]=secondhalf,out[15]=firstbit);//no

Or8Way(in=firsthalf,out=firstor);

Or8Way(in=secondhalf,out=secondor);

Or(a=firstor,b=secondor,out=outor);

Xor(a=outor, b=true, out=zr);//zr

And(a=firstbit,b=true,out=ng);//ng

}

**ADD16.HDL**

CHIP Add16 {

IN a[16], b[16];

OUT out[16];

PARTS:

HalfAdder(a=a[0],b=b[0],sum=out[0],carry=carry1);

FullAdder(a=a[1],b=b[1],c=carry1,sum=out[1],carry=carry2);

FullAdder(a=a[2],b=b[2],c=carry2,sum=out[2],carry=carry3);

FullAdder(a=a[3],b=b[3],c=carry3,sum=out[3],carry=carry4);

FullAdder(a=a[4],b=b[4],c=carry4,sum=out[4],carry=carry5);

FullAdder(a=a[5],b=b[5],c=carry5,sum=out[5],carry=carry6);

FullAdder(a=a[6],b=b[6],c=carry6,sum=out[6],carry=carry7);

FullAdder(a=a[7],b=b[7],c=carry7,sum=out[7],carry=carry8);

FullAdder(a=a[8],b=b[8],c=carry8,sum=out[8],carry=carry9);

FullAdder(a=a[9],b=b[9],c=carry9,sum=out[9],carry=carry10);

FullAdder(a=a[10],b=b[10],c=carry10,sum=out[10],carry=carry11);

FullAdder(a=a[11],b=b[11],c=carry11,sum=out[11],carry=carry12);

FullAdder(a=a[12],b=b[12],c=carry12,sum=out[12],carry=carry13);

FullAdder(a=a[13],b=b[13],c=carry13,sum=out[13],carry=carry14);

FullAdder(a=a[14],b=b[14],c=carry14,sum=out[14],carry=carry15);

Xor(a=a[15],b=b[15],out=a15Xorb15);

Xor(a=a15Xorb15,b=carry15,out=out[15]);

}

**FULLADDER.HDL**

CHIP FullAdder {

IN a, b, c; // 1-bit inputs

OUT sum, // Right bit of a + b + c

carry; // Left bit of a + b + c

PARTS:

Xor(a=a,b=b,out=aXorb);

Xor(a=aXorb,b=c,out=sum);

Not(in=b,out=notb);

And(a=a,b=notb,out=aAndNotb);

And(a=aAndNotb,b=c,out=aAndNotbAndc);

Not(in=c,out=notc);

And(a=a,b=b,out=aAndb);

And(a=aAndb,b=notc,out=aAndbAndNotc);

And(a=b,b=c,out=bAndc);

Or(a=aAndbAndNotc,b=aAndNotbAndc,out=firstOr);

Or(a=firstOr,b=bAndc,out=carry);

}

**HALFADDER.HDL**

CHIP HalfAdder {

IN a, b; // 1-bit inputs

OUT sum, // Right bit of a + b

carry; // Left bit of a + b

PARTS:

Xor(a=a,b=b,out=sum);

And(a=a,b=b,out=carry);

}