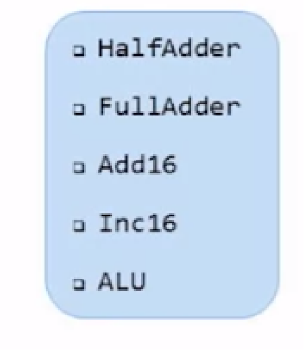
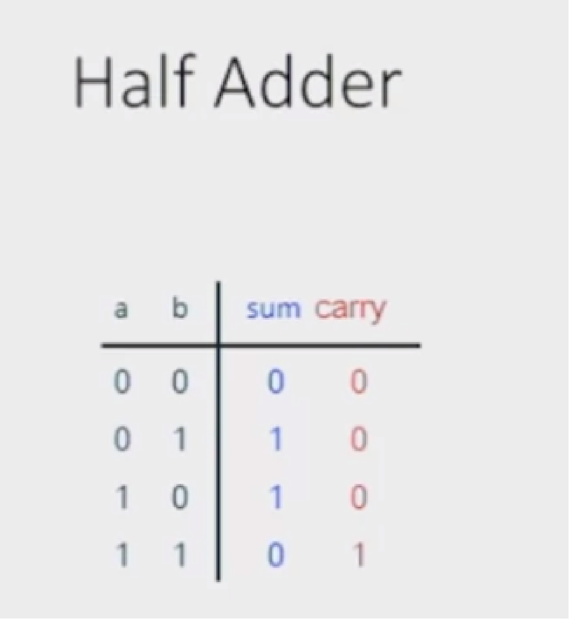
project2



- HalfAdder
- FullAdder
- Add16
- Inc16
- ALU

HalfAdder

真值表：



Half Adder

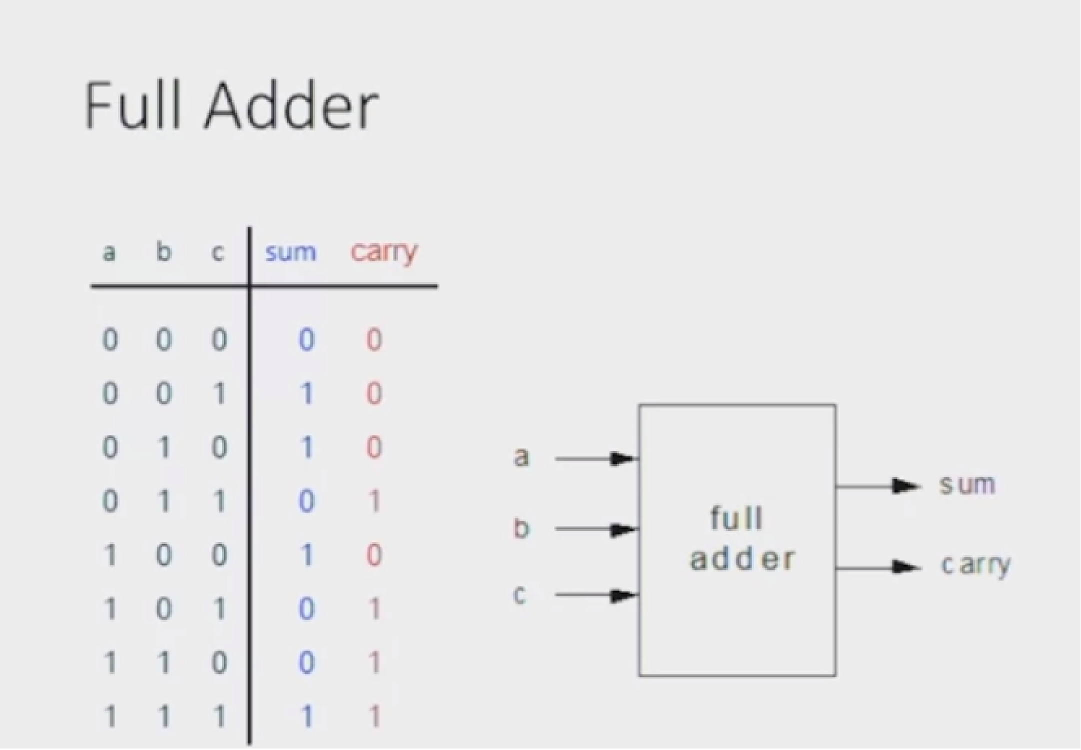
a	b	sum	carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

可以看出sum 是Xor(a,b)
而carry是 And(a, b)

```
// This file is part of www.nand2tetris.org
// and the book "The Elements of Computing Systems"
// by Nisan and Schocken, MIT Press.
// File name: projects/02/HalfAdder.hdl
/**
 * Computes the sum of two bits.
 */
CHIP HalfAdder {
    IN a, b;    // 1-bit inputs
    OUT sum,    // Right bit of a + b
        carry;  // Left bit of a + b
    PARTS:
    // Put you code here:
    And (a=a , b=b, out=carry );
    Xor (a=a, b=b, out =sum);
}
```

FullAdder

真值表:



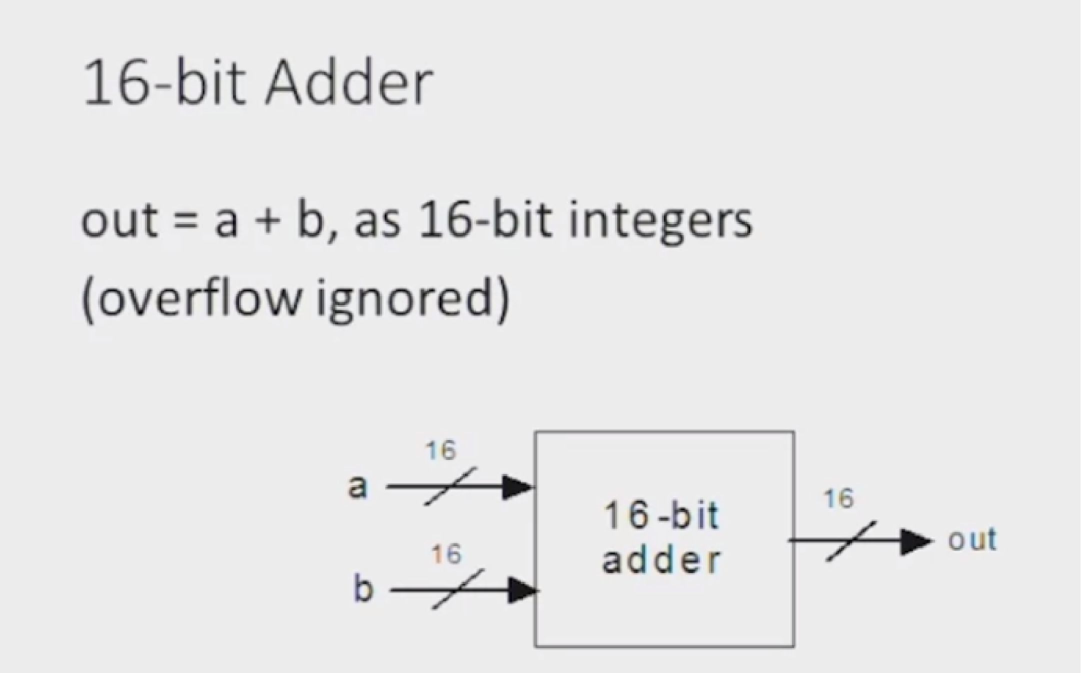
Full Adder

a	b	c	sum	carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

全加器就是有了一个进位a 即a+b+c
可以考虑先将其HalfAdder(b,c)后，得到sum1 和compare1
再HalfAdder(a,sum1) 得到sum，compare2
再HalfAdder(compare1,compare2)得到sum = compare

```
// This file is part of www.nand2tetris.org
// and the book "The Elements of Computing Systems"
// by Nisan and Schocken, MIT Press.
// File name: projects/02/FullAdder.hdl
/**
 * Computes the sum of three bits.
 */
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:
    // Put you code here:
    HalfAdder( a =b ,b = c,sum = sum1 ,carry = carry1);
    HalfAdder( a =a ,b = sum1,sum = sum ,carry = carry2);
    HalfAdder( a =carry1 ,b = carry2,sum =carry ,carry = nothing);
}
```

Add16



16-bit Adder

out = a + b, as 16-bit integers
(overflow ignored)

```
// This file is part of www.nand2tetris.org
// and the book "The Elements of Computing Systems"
// by Nisan and Schocken, MIT Press.
// File name: projects/02/Adder16.hdl
/**
 * Adds two 16-bit values.
 * The most significant carry bit is ignored.
 */
CHIP Add16 {
    IN a[16], b[16];
    OUT out[16];
    PARTS:
   // Put you code here:
   HalfAdder(a = a[0],b= b[0],sum=out[0],carry = carry1);
   FullAdder(a = carry1 ,b= a[1],c =b[1],sum=out[1],carry =carry2);
   FullAdder(a = carry2 ,b= a[2],c =b[2],sum=out[2],carry =carry3);
   FullAdder(a = carry3 ,b= a[3],c =b[3],sum=out[3],carry =carry4);
   FullAdder(a = carry4 ,b= a[4],c =b[4],sum=out[4],carry =carry5);
   FullAdder(a = carry5 ,b= a[5],c =b[5],sum=out[5],carry =carry6);
   FullAdder(a = carry6 ,b= a[6],c =b[6],sum=out[6],carry =carry7);
   FullAdder(a = carry7 ,b= a[7],c =b[7],sum=out[7],carry =carry8);
   FullAdder(a = carry8 ,b= a[8],c =b[8],sum=out[8],carry =carry9);
   FullAdder(a = carry9 ,b= a[9],c =b[9],sum=out[9],carry =carry10);
   FullAdder(a = carry10 ,b= a[10],c =b[10],sum=out[10],carry =carry11);
   FullAdder(a = carry11 ,b= a[11],c =b[11],sum=out[11],carry =carry12);
   FullAdder(a = carry12 ,b= a[12],c =b[12],sum=out[12],carry =carry13);
   FullAdder(a = carry13 ,b= a[13],c =b[13],sum=out[13],carry =carry14);
   FullAdder(a = carry14 ,b= a[14],c =b[14],sum=out[14],carry =carry15);
```

```
    FullAdder(a = carry15 ,b= a[15],c =b[15],sum=out[15],carry =nothing);
}
```

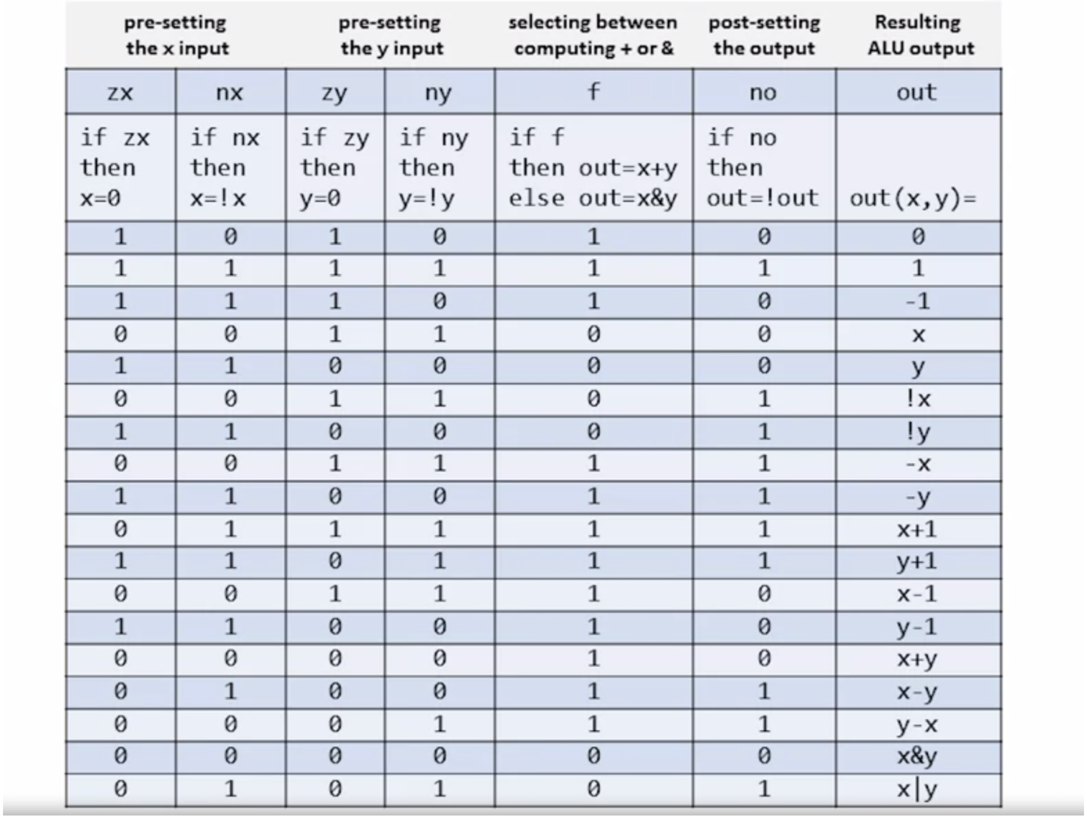
Inc16

就是加一，只需要注意语法上的实现

```
// This file is part of www.nand2tetris.org
// and the book "The Elements of Computing Systems"
// by Nisan and Schocken, MIT Press.
// File name: projects/02/Inc16.hdl
/**
 * 16-bit incrementer:
 * out = in + 1 (arithmetic addition)
 */
CHIP Inc16 {
    IN in[16];
    OUT out[16];
    PARTS:
   // Put you code here:
   Add16(a = in,b[0] = true,b[1..15]=false,out=out);
}
```

ALU

我遇到的问题是内部的接口似乎无法切片（[0..3]这种操作）？



pre-setting the x input		pre-setting the y input		selecting between computing + or &	post-setting the output	Resulting ALU output
zx	nx	zy	ny	f	no	out
if zx then x=0	if nx then x=!x	if zy then y=0	if ny then y=!y	if f then out=x+y else out=x&y	if no then out=!out	out(x,y)=
1	0	1	0	1	0	0
1	1	1	1	1	1	1
1	1	1	0	1	0	-1
0	0	1	1	0	0	x
1	1	0	0	0	0	y
0	0	1	1	0	1	!x
1	1	0	0	0	1	!y
0	0	1	1	1	1	-x
1	1	0	0	1	1	-y
0	1	1	1	1	1	x+1
1	1	0	1	1	1	y+1
0	0	1	1	1	0	x-1
1	1	0	0	1	0	y-1
0	0	0	0	1	0	x+y
0	1	0	0	1	1	x-y
0	0	0	1	1	1	y-x
0	0	0	0	0	0	x&y
0	1	0	1	0	1	x|y

```
// This file is part of www.nand2tetris.org
// and the book "The Elements of Computing Systems"
// by Nisan and Schocken, MIT Press.
// File name: projects/02/ALU.hdl
/**
 * The ALU (Arithmetic Logic Unit).
 * Computes one of the following functions:
 * x+y, x-y, y-x, 0, 1, -1, x, y, -x, -y, !x, !y,
 * x+1, y+1, x-1, y-1, x&y, x|y on two 16-bit inputs,
 * according to 6 input bits denoted zx,nx,zy,ny,f,no.
 * In addition, the ALU computes two 1-bit outputs:
 * if the ALU output == 0, zr is set to 1; otherwise zr is set to 0;
 * if the ALU output < 0, ng is set to 1; otherwise ng is set to 0.
 */
// Implementation: the ALU logic manipulates the x and y inputs
// and operates on the resulting values, as follows:
// if (zx == 1) set x = 0        // 16-bit constant
// if (nx == 1) set x = !x       // bitwise not
// if (zy == 1) set y = 0        // 16-bit constant
// if (ny == 1) set y = !y       // bitwise not
// if (f == 1)  set out = x + y  // integer 2's complement addition
// if (f == 0)  set out = x & y  // bitwise and
// if (no == 1) set out = !out   // bitwise not
// if (out == 0) set zr = 1
// if (out < 0) set ng = 1
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:
   // Put you code here:
   //process zx nx
   Mux16(a=x ,b= false,sel = zx,out = xval);
   Not16(in=xval, out=notx);
   Mux16(a=xval ,b= notx,sel = nx,out = lastx);
   //process zy ny
   Mux16(a=y ,b= false,sel = zy,out = yval);
   Not16(in = yval,out = noty);
   Mux16(a=yval ,b= noty,sel = ny,out = lasty);
   //pocsee f
   And16(a=lastx,b=lasty,out = Andf);
```

```
    Add16(a=lastx,b=lasty,out = Addf);
    Mux16(a =Andf,b = Addf,sel=f,out = outMux);
    //process no
    Not16(in = outMux,out = outNot);
    Mux16(a =outMux,b=outNot,sel =  no,out =out1);
    //set zr
    Or16Way(in=out1,out=orlast);
    Mux(a=true,b=false,sel=orlast,out =zr);

    //set ng
    IsNeg(in=out1,out =ng);
    Or16(a=out1,b=false,out=out);
}
```