project1

在本课程中我们将以Nand（与非门）为前提来建造我们的芯片
就像课程中说的那样：

> 上帝给了我们nand

与非门真值表

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

任务如下：

Project 1

Elementary logic gates	16-bit variants	Multi-way variants
Not	Not16	Or8Way
And	And16	Mux4Way16
Or	Or16	Mux8Way16
Xor	Mux16	DMux4Way
Mux		DMux8Way
DMux		

逻辑门

Not

我们先来构建最基本的非门

```
//Not.hdl
CHIP Not {
    IN in;
    OUT out;
    PARTS:
    // Put your code here:
    Nand (a =in,b =in,out =out);
}
```

非门的真值表

输入A	输出Y
0	1
1	0

And

很明显And=Not(Nand(a,b))

输入A	输入B	输出Y
0	0	0
0	1	0
1	0	0
1	1	1

```
// 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/01/And.hdl
/**
 * And gate:
 * out = 1 if (a == 1 and b == 1)
 *       0 otherwise
 */
CHIP And {
    IN a, b;
    OUT out;
    PARTS:
    // Put your code here:
    Nand (a = a,b =b,out = Nand1 );
    Not  (in = Nand1 ,out =out );
}

```

Or

a Or b =Not (Not(a) and Not(b))

或门的真值表

输入A	输入B	输出F
0	0	0
0	1	1
1	0	1
1	1	1

```
// 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/01/Or.hdl
 /**
 * Or gate:
 * out = 1 if (a == 1 or b == 1)
 *       0 otherwise
 */
CHIP Or {
    IN a, b;
    OUT out;
    PARTS:
    // Put your code here:
    Not (in =a,out =Nota);
    Not (in =b,out =Notb);
    And (a= Nota,b =Notb ,out =And1);
    Not (in = And1, out =out);
}

```

Xor

就直接用课堂上的demo了
真值表：

A	B	输出Y
0	0	0
0	1	1
1	0	1
1	1	0

```
// 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/01/Xor.hdl
/**
 * Exclusive-or gate:
 * out = not (a == b)
 */
CHIP Xor {
    IN a, b;
    OUT out;
    PARTS:
    Not (in=a, out = nota);
    Not (in=b, out = notb);
    And (a=a, b=notb, out=w1);
    And (a=nota, b=b, out=w2);
    Or  (a=w1, b=w2, out=out);
}
```

Mux

伪代码：

```
/**
 * Multiplexor:
 * out = a if sel == 0
 *       b otherwise
 */
```

Truth table:

a	b	sel	out
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

根据真值表写出如下表达式：

(a And Not(b) And Not(sel)) Or (a And b And Not(sel)) Or (Not(a) And b And sel) Or (a And b And sel)

(a And Not(b) And Not(sel)) Or (a And b And Not(sel))

=(结合律) (a And Not(sel)) and(b or Notb)= a And Not(sel)

(Not(a) And b And sel) Or (a And b And sel)

= (同理) b And sel

=> a And Not(sel) Or b And sel

```
// 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/01/Mux.hdl
/**
 * Multiplexor:
 * out = a if sel == 0
 *       b otherwise
 */
CHIP Mux {
    IN a, b, sel;
    OUT out;
    PARTS:
    // Put your code here:
    Not (in = sel, out =Notsel);
    And (a = a, b=Notsel , out
=And1);
    And (a = b, b= sel, out = And2);
    Or(a = And1,b = And2,out = out);

}
```

Dmux

in	sel	a	b
0	0	0	0
0	1	0	0
1	0	1	0
1	1	0	1

```
/**
 * Demultiplexor:
 * {a, b} = {in, 0} if sel == 0
 *          {0, in} if sel == 1
 */
```

这个的思路是将ab分开，即if sel ==0 ： a=in;

else if sel==1 : a=0;

```
// 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/01/DMux.hdl
/**
 * Demultiplexor:
 * {a, b} = {in, 0} if sel == 0
 *          {0, in} if sel == 1
 */
CHIP DMux {
    IN in, sel;
    OUT a, b;
    PARTS:
    // Put your code here:
    Not (in = sel ,out = Notsel);
    Mux(a = in,b = Notsel ,sel = sel,out =a );
    Mux(a = sel,b = in ,sel = sel,out =b );
}

```

16位的逻辑门（variants）

And16

简单粗暴And16次：

a	b	out
0000000000000000	0000000000000000	0000000000000000
0000000000000000	1111111111111111	0000000000000000
1111111111111111	1111111111111111	1111111111111111
1010101010101010	0101010101010101	0000000000000000
0011110011000011	0000111111110000	0000110011000000
0001001000110100	1001100001110110	0001000000110100

```
// 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/01/And16.hdl
/**
 * 16-bit bitwise And:
 * for i = 0..15: out[i] = (a[i] and b[i])
```

```
 */
CHIP And16 {
    IN a[16], b[16];
    OUT out[16];
    PARTS:
    // Put your code here:
    And (a = a[0],b =b[0],out =out[0]);
    And (a = a[1],b =b[1],out =out[1]);
    And (a = a[2],b =b[2],out =out[2]);
    And (a = a[3],b =b[3],out =out[3]);
    And (a = a[4],b =b[4],out =out[4]);
    And (a = a[5],b =b[5],out =out[5]);
    And (a = a[6],b =b[6],out =out[6]);
    And (a = a[7],b =b[7],out =out[7]);
    And (a = a[8],b =b[8],out =out[8]);
    And (a = a[9],b =b[9],out =out[9]);
    And (a = a[10],b =b[10],out =out[10]);
    And (a = a[11],b =b[11],out =out[11]);
    And (a = a[12],b =b[12],out =out[12]);
    And (a = a[13],b =b[13],out =out[13]);
    And (a = a[14],b =b[14],out =out[14]);
    And (a = a[15],b =b[15],out =out[15]);
}
```

Not16

同理

in	out
0000000000000000	1111111111111111
1111111111111111	0000000000000000
1010101010101010	0101010101010101
0011110011000011	1100001100111100
0001001000110100	1110110111001011

```
// 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/01/Not16.hdl
/**
 * 16-bit Not:
 * for i=0..15: out[i] = not in[i]
 */
CHIP Not16 {
    IN in[16];
    OUT out[16];
    PARTS:
    // Put your code here:
    Not(in = in[0],out = out[0]);
    Not(in = in[1],out = out[1]);
    Not(in = in[2],out = out[2]);
    Not(in = in[3],out = out[3]);
```

```
    Not(in = in[4],out = out[4]);
    Not(in = in[5],out = out[5]);
    Not(in = in[6],out = out[6]);
    Not(in = in[7],out = out[7]);
    Not(in = in[8],out = out[8]);
    Not(in = in[9],out = out[9]);
    Not(in = in[10],out = out[10]);
    Not(in = in[11],out = out[11]);
    Not(in = in[12],out = out[12]);
    Not(in = in[13],out = out[13]);
    Not(in = in[14],out = out[14]);
    Not(in = in[15],out = out[15]);
}
```

Or16

a	b	out
0000000000000000	0000000000000000	0000000000000000
0000000000000000	1111111111111111	1111111111111111
1111111111111111	1111111111111111	1111111111111111
1010101010101010	0101010101010101	1111111111111111
0011110011000011	0000111111110000	0011111111110011
0001001000110100	1001100001110110	1001101001110110

```
    And (a = a[0],b =b[0],out =out[0]);
    And (a = a[1],b =b[1],out =out[1]);
    And (a = a[2],b =b[2],out =out[2]);
    And (a = a[3],b =b[3],out =out[3]);
    And (a = a[4],b =b[4],out =out[4]);
    And (a = a[5],b =b[5],out =out[5]);
    And (a = a[6],b =b[6],out =out[6]);
    And (a = a[7],b =b[7],out =out[7]);
    And (a = a[8],b =b[8],out =out[8]);
    And (a = a[9],b =b[9],out =out[9]);
    And (a = a[10],b =b[10],out =out[10]);
    And (a = a[11],b =b[11],out =out[11]);
    And (a = a[12],b =b[12],out =out[12]);
    And (a = a[13],b =b[13],out =out[13]);
    And (a = a[14],b =b[14],out =out[14]);
    And (a = a[15],b =b[15],out =out[15]);
```

Mux16

a	b	sel	out
0000000000000000	0000000000000000	0	0000000000000000
0000000000000000	0000000000000000	1	0000000000000000
0000000000000000	0001001000110100	0	0000000000000000
0000000000000000	0001001000110100	1	0001001000110100
1001100001110110	0000000000000000	0	1001100001110110
1001100001110110	0000000000000000	1	0000000000000000
1010101010101010	0101010101010101	0	1010101010101010
1010101010101010	0101010101010101	1	0101010101010101

```
  // 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/01/Mux16.hdl
/**
 * 16-bit multiplexor:
 * for i = 0..15 out[i] = a[i] if sel == 0
 *                        b[i] if sel == 1
 */
CHIP Mux16 {
    IN a[16], b[16], sel;
    OUT out[16];
    PARTS:
    // Put your code here:
    Mux(a = a[0],b =b[0],sel = sel,out =out[0]);
    Mux(a = a[1],b =b[1],sel = sel,out =out[1]);
    Mux(a = a[2],b =b[2],sel = sel,out =out[2]);
    Mux(a = a[3],b =b[3],sel = sel,out =out[3]);
    Mux(a = a[4],b =b[4],sel = sel,out =out[4]);
    Mux(a = a[5],b =b[5],sel = sel,out =out[5]);
    Mux(a = a[6],b =b[6],sel = sel,out =out[6]);
    Mux(a = a[7],b =b[7],sel = sel,out =out[7]);
    Mux(a = a[8],b =b[8],sel = sel,out =out[8]);
    Mux(a = a[9],b =b[9],sel = sel,out =out[9]);
    Mux(a = a[10],b =b[10],sel = sel,out =out[10]);
    Mux(a = a[11],b =b[11],sel = sel,out =out[11]);
    Mux(a = a[12],b =b[12],sel = sel,out =out[12]);
    Mux(a = a[13],b =b[13],sel = sel,out =out[13]);
    Mux(a = a[14],b =b[14],sel = sel,out =out[14]);
    Mux(a = a[15],b =b[15],sel = sel,out =out[15]);
}

```

多路逻辑门

Or8Way

定义：

/**

 * 8-way Or:

 * out = (in[0] or in[1] or ... or in[7])

 */

真值表

in	out
00000000	0
11111111	1
00010000	1
00000001	1
00100110	1

```
// 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/01/Or8Way.hdl
/**
 * 8-way Or:
 * out = (in[0] or in[1] or ... or in[7])
 */
CHIP Or8Way {
    IN in[8];
    OUT out;
    PARTS:
    // Put your code here:
    Or (a=in[0] , b=in[1],out = out1);
    Or (a=out1 , b=in[2],out = out2);
    Or (a=out2 , b=in[3],out = out3);
    Or (a=out3 , b=in[4],out = out4);
    Or (a=out4 , b=in[5],out = out5);
    Or (a=out5 , b=in[6],out = out6);
    Or (a=out6 , b=in[7],out = out);
}
```

Mux4way16

target:

```
/**
 * 4-way 16-bit multiplexor:
 * out = a if sel == 00
 *       b if sel == 01
 *       c if sel == 10
 *       d if sel == 11
 */
```

这个真值表不太好截图:
大致说一下思路，还是把目的一分为二 先对sel的一位进行判断
if sel[1] ==0
  out={a,b}
else if sel[1] ==1
  out ={c,d}
进行分类，在对输出结果进行sel[0]进行分类

```
// 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/01/Mux4Way16.hdl
/**
 * 4-way 16-bit multiplexor:
 * out = a if sel == 00
 *       b if sel == 01
 *       c if sel == 10
 *       d if sel == 11
 */
```

```
CHIP Mux4Way16 {
    IN a[16], b[16], c[16], d[16], sel[2];
    OUT out[16];
    PARTS:
    // Put your code here:
    Mux16(a=a ,b=c ,sel=sel[1] ,out=out1 );
   //Mux16(a= a[0..15],b= c[0..15],sel =sel[1],out =m1 );
    Mux16(a=b ,b=d ,sel=sel[1] ,out=out2 );
    Mux16(a=out1 ,b=out2 ,sel=sel[0] ,out=out );

}
```

Mux8Way16

这里就使用Mux4way16，来分别从a,b,c,d和e,f,g,h根据sel[0..1]选出相应的位，之后根据最高位再进行一次Mux16

```
// 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/01/Mux8Way16.hdl
/**
 * 8-way 16-bit multiplexor:
 * out = a if sel == 000
 *       b if sel == 001
 *       etc.
 *       h if sel == 111
 */
CHIP Mux8Way16 {
    IN a[16], b[16], c[16], d[16],
       e[16], f[16], g[16], h[16],
       sel[3];
    OUT out[16];
    PARTS:
    // Put your code here:
    Mux4Way16(a=a ,b= b,c= c,d=d ,sel= sel[0..1],out=out1 );
    Mux4Way16(a=e ,b=f ,c=g, d=h, sel= sel[0..1],out = out2 );
    Mux16(a = out1, b = out2 ,sel=sel[2] , out =out);
}
```

DMux4Way

truth table:

in	sel	a	b	c	d
0	00	0	0	0	0
0	01	0	0	0	0
0	10	0	0	0	0
0	11	0	0	0	0
1	00	1	0	0	0
1	01	0	1	0	0
1	10	0	0	1	0
1	11	0	0	0	1

伪代码：

```
/**
 * 4-way demultiplexor:
 * {a, b, c, d} = {in, 0, 0, 0} if sel == 00
 *                {0, in, 0, 0} if sel == 01
 *                {0, 0, in, 0} if sel == 10
 *                {0, 0, 0, in} if sel == 11
 */
```

```
// 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/01/DMux4Way.hdl
/**
 * 4-way demultiplexor:
 * {a, b, c, d} = {in, 0, 0, 0} if sel == 00
 *                {0, in, 0, 0} if sel == 01
 *                {0, 0, in, 0} if sel == 10
 *                {0, 0, 0, in} if sel == 11
 */
CHIP DMux4Way {
    IN in, sel[2];
    OUT a, b, c, d;
    PARTS:
    // Put your code here:
    DMux(in = in,sel=sel[0], a =out1, b =out2);
    DMux(in = out1,sel = sel[1],a=a,b=c);
    DMux(in = out2,sel = sel[1],a=b,b=d);
}
```

DMux16Way

truth table:

in	sel	a	b	c	d	e	f	g	h
0	000	0	0	0	0	0	0	0	0
0	001	0	0	0	0	0	0	0	0
0	010	0	0	0	0	0	0	0	0
0	011	0	0	0	0	0	0	0	0
0	100	0	0	0	0	0	0	0	0
0	101	0	0	0	0	0	0	0	0
0	110	0	0	0	0	0	0	0	0
0	111	0	0	0	0	0	0	0	0
1	000	1	0	0	0	0	0	0	0
1	001	0	1	0	0	0	0	0	0
1	010	0	0	1	0	0	0	0	0
1	011	0	0	0	1	0	0	0	0
1	100	0	0	0	0	1	0	0	0
1	101	0	0	0	0	0	1	0	0
1	110	0	0	0	0	0	0	1	0
1	111	0	0	0	0	0	0	0	1

伪代码

```
/**
 * 8-way demultiplexor:
 * {a, b, c, d, e, f, g, h} = {in, 0, 0, 0, 0, 0, 0, 0} if sel == 000
```

```
*                   {0, in, 0, 0, 0, 0, 0, 0} if sel == 001
*                   etc.
*                   {0, 0, 0, 0, 0, 0, 0, in} if sel == 111
*/
```

```
// 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/01/DMux8Way.hdl
/**
 * 8-way demultiplexor:
 * {a, b, c, d, e, f, g, h} = {in, 0, 0, 0, 0, 0, 0, 0} if sel == 000
 *                            {0, in, 0, 0, 0, 0, 0, 0} if sel == 001
 *                            etc.
 *                            {0, 0, 0, 0, 0, 0, 0, in} if sel == 111
 */
CHIP DMux8Way {
    IN in, sel[3];
    OUT a, b, c, d, e, f, g, h;
    PARTS:
    // Put your code here:
    DMux4Way(in = in,sel = sel[0..1],a = out1, b= out2, c=out3, d=out4);
    DMux(in = out1,sel = sel[2],a=a,b=e);
    DMux(in = out2,sel = sel[2],a=b,b=f);
    DMux(in = out3,sel = sel[2],a=c,b=g);
    DMux(in = out4,sel = sel[2],a=d,b=h);
}
```