



COM-ORG

W2 : Project-2 Overview

31110321 Computer Organization

สำหรับนักศึกษาชั้นปีที่ 3 สาขาวิชาวิศวกรรมคอมพิวเตอร์

ทรงฤทธิ์ กิตติศรีวรพันธุ์

songrit@npu.ac.th

สาขาวิชาวิศวกรรมคอมพิวเตอร์
มหาวิทยาลัยนครพนม

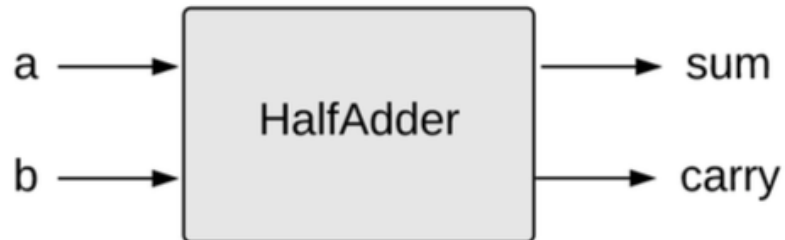
Outline

- Chipset : ใช้ชิปจาก Project-1
- Goal : สร้างชิปเซตต่อไปนี้

- HalfAdder
- FullAdder
- Add16
- Inc16
- ALU

ใช้ชิปเซตจาก Project-1 ประกอบเป็นชิปเซตใหม่

Half Adder



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

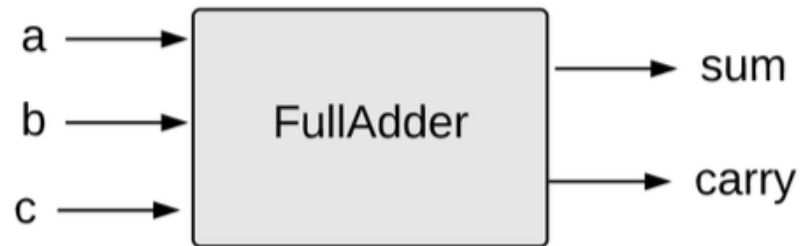
HalfAdder.hdl

```
/** Computes the sum of two bits. */  
  
CHIP HalfAdder {  
    IN a, b;  
    OUT sum, carry;  
  
    PARTS:  
    // Put your code here:  
}
```

- คำแนะนำ

- สามารถใช้ลอจิกเกตพื้นฐานสร้าง Half Adder ได้

Full Adder



FullAdder.hdl

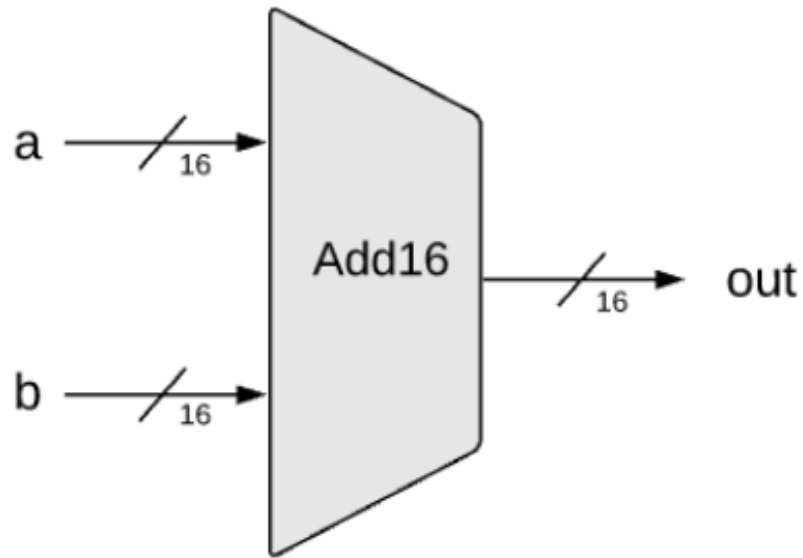
```
/** Computes the sum of three bits. */  
  
CHIP HalfAdder {  
    IN a, b, c;  
    OUT sum, carry;  
  
    PARTS:  
    // Put your code here:  
}
```

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

- คำแนะนำ

- สามารถใช้ 2 Half Adder
- สร้าง Full adder

16-bit adder



Add16.hdl

```
/*
 * Adds two 16-bit, two's-complement values.
 * The most-significant carry bit is ignored.
 */

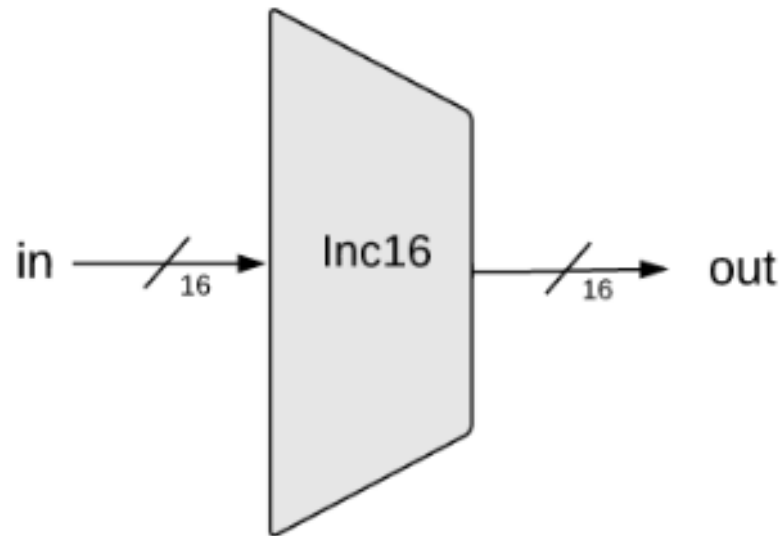
CHIP Add16 {
    IN a[16], b[16];
    OUT out[16];

    PARTS:
    // Put your code here:
}
```

- คำแนะนำ

- ใช้ n-bit adder ใช้สร้าง n-bit full-adder
- Carry bit เป็นลำดับต่อเนื่องจากผลของ carry bit จากบิตขวาไปบิตซ้าย
- ไม่สนใจ MSB carry bit

16-bit incrementor



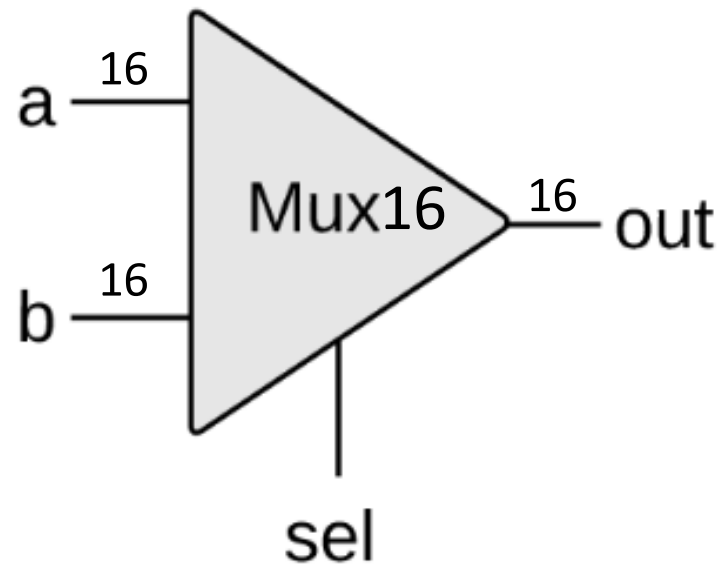
- คำแนะนำ

- ผลลัพธ์ $out = in + 1_b$
- พิจารณาระบบ Inc2 (2บิต)
- มีกรณีสืบ carry จากบิตทางขวา
- มีกรณีส่งต่อ carry บิตไปบิตทางซ้าย

```
/*  
 * Outputs in + 1.  
 * The most-significant carry bit is ignored.  
 */  
  
CHIP Inc16 {  
    IN in[16];  
    OUT out[16];  
  
    PARTS:  
    // Put you code here:  
}
```

Chipset (composite)

- Mux16, Not16, And16, Or8Way



sel	out
0	a
1	b

Not16, And16, Or8Way

Chip name: Not16

Inputs: in[16] // a 16-bit pin

Outputs: out[16]

Function: For i=0..15 out[i]=Not(in[i]).

Not16 -> flip-bit

Chip name: And16

Inputs: a[16], b[16]

Outputs: out[16]

Function: For i=0..15 out[i]=And(a[i],b[i]).

And16 คู่กัน

Chip name: Or8Way

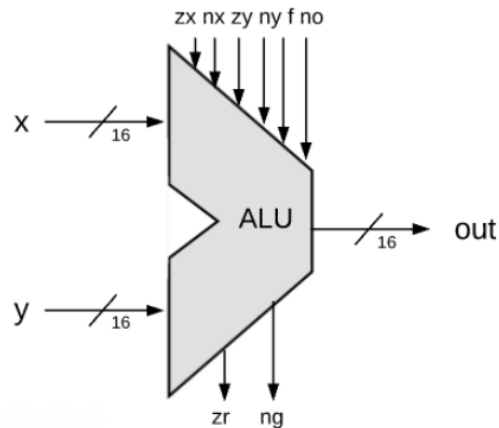
Inputs: in[8]

Outputs: out

Function: out=Or(in[0],in[1],...,in[7]).

out=0 กรณีเดียว

ALU



- คำแนะนำ

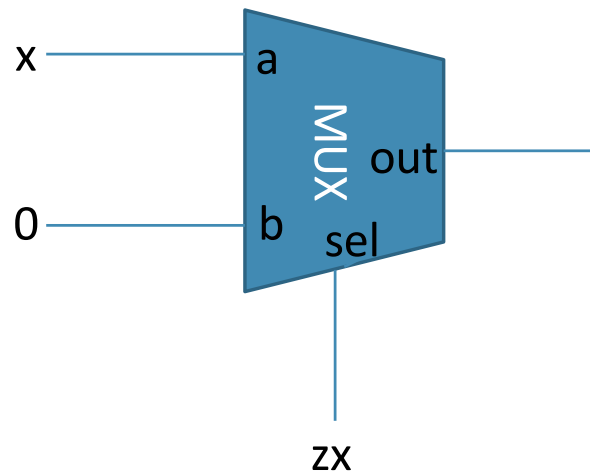
- สร้าง blocks : Add16 และใช้ลอจิกเกตจาก Project-1
- สามารถสร้าง ALU ได้โดยใช้ HDL code ประมาณ 20 บรรทัด
- ใช้ทุกชิปเซตที่ทำมาช่วยให้โค้ดสั้น

ALU.hdl

```
/** The ALU. */
// Manipulates the x and y inputs as follows:
// if (zx == 1) sets x = 0           // 16-bit true constant
// if (nx == 1) sets x = !x         // bitwise Not
// if (zy == 1) sets y = 0           // 16-bit true constant
// if (ny == 1) sets y = !y         // bitwise Not
// if (f == 1) sets out = x + y     // int. 2's-complement addition
// if (f == 0) sets out = x & y     // bitwise And
// if (no == 1) sets out = !out      // bitwise Not
// if (out == 0) sets zr = 1         // 1-bit true constant
// if (out < 0) sets ng = 1          // 1-bit true constant
...
```

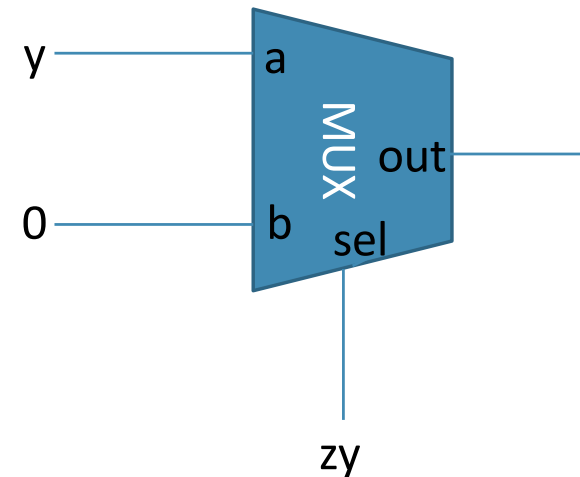
ALU components : zx, zy

- ใช้ Mux สำหรับ zx



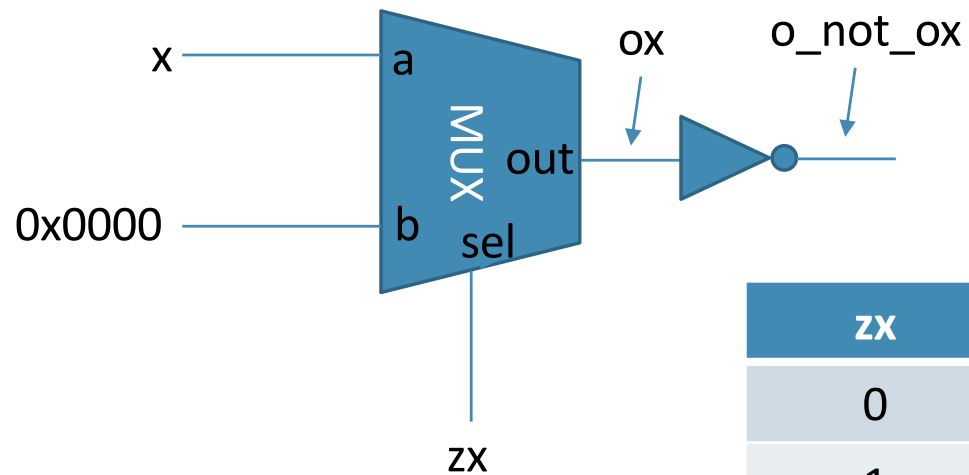
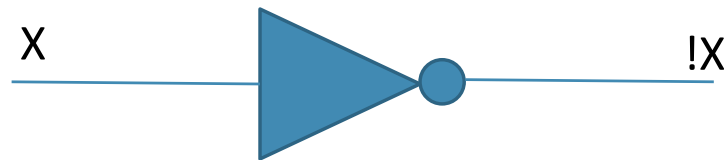
sel	out
0	a
1	b

- ใช้ Mux สำหรับ zy



ALU components : nx

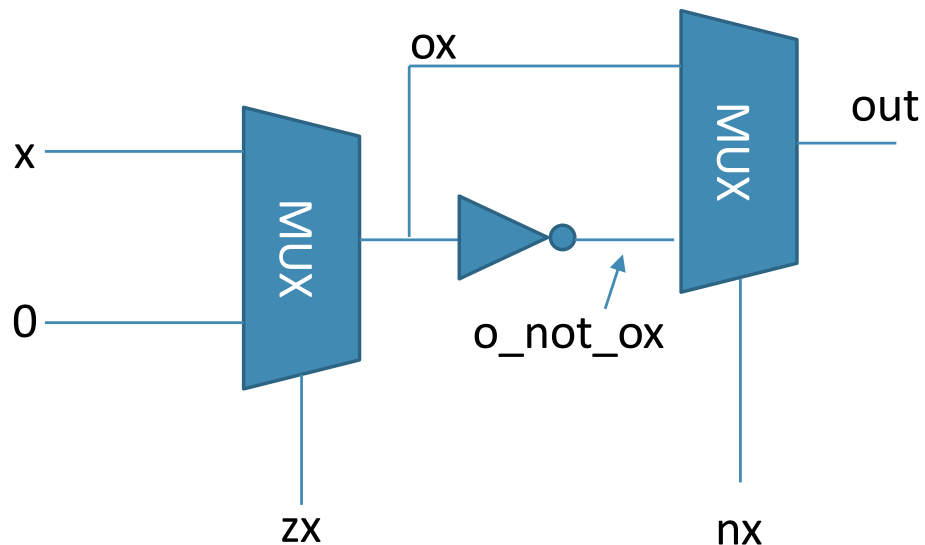
- Negative X



zx	ox	o_not_ox
0	x	!x
1	0x0000	0x1111

ALU components : nx

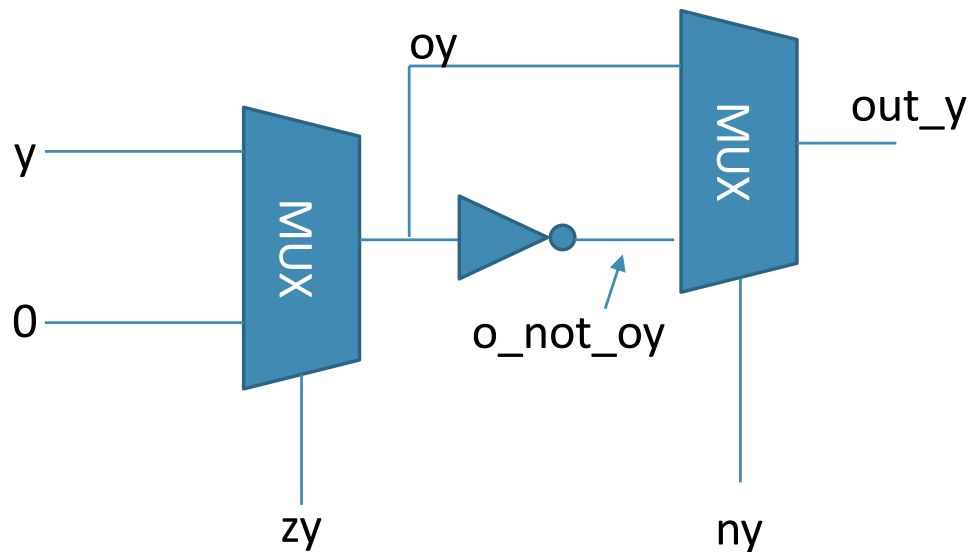
- ใช้ Mux สำหรับ zx, nx



zx	nx	out
0	0	x
0	1	!x
1	0	0x0000
1	1	0x1111

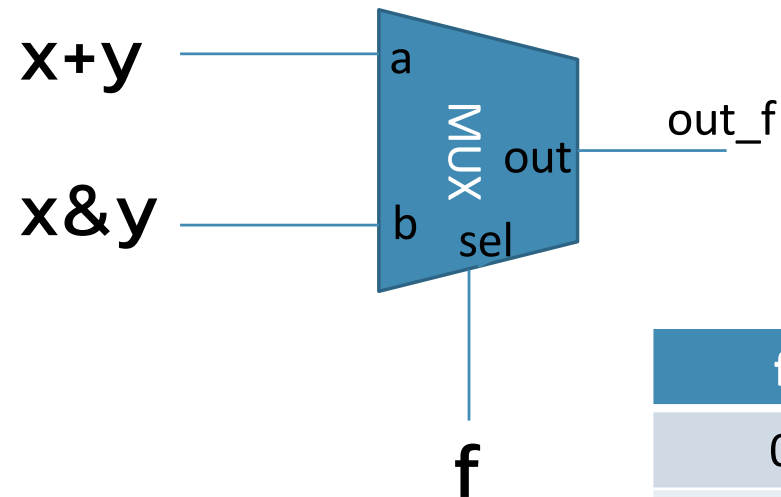
ALU components : zy, ny

- เชื่อมเดียวกับ zx, nx



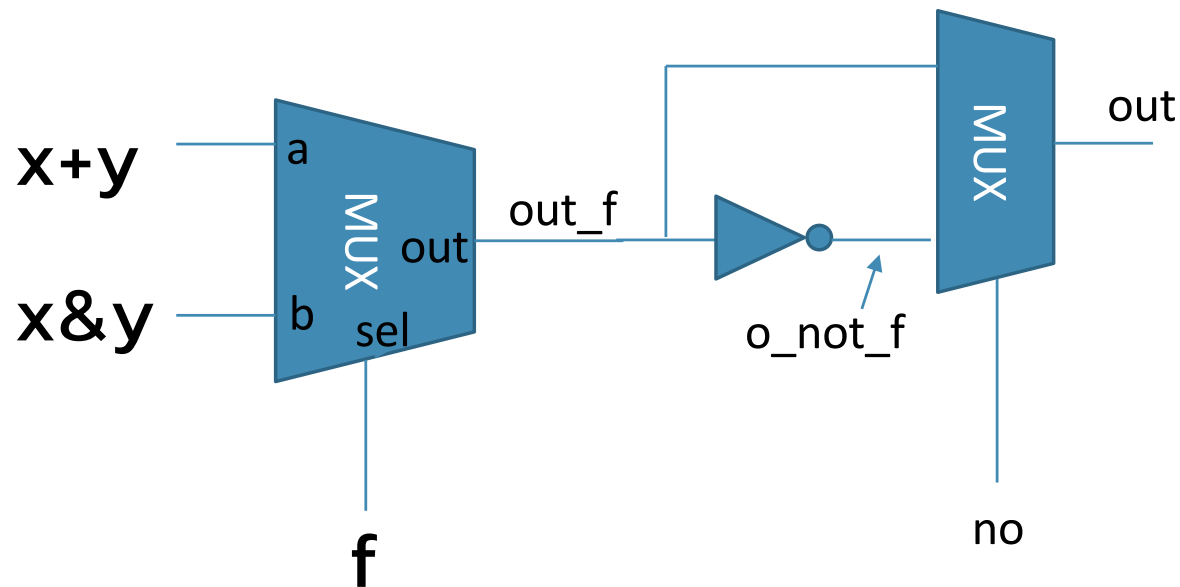
zy	ny	out_y
0	0	y
0	1	!y
1	0	0x0000
1	1	0x1111

ALU : $x+y$ หรือ $x \& y$



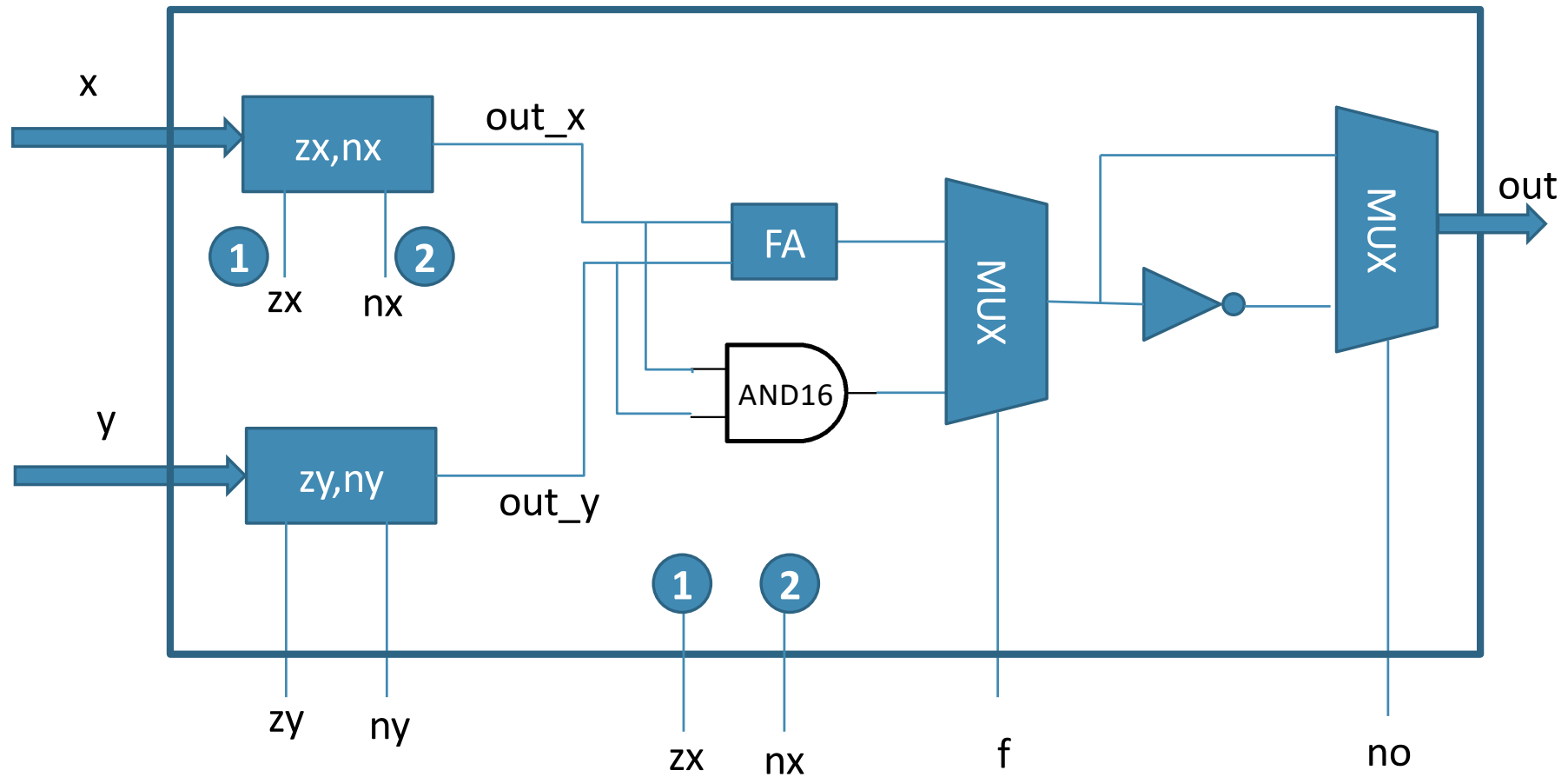
f	out_f
0	$x+y$
1	$x \& y$

ALU : $x+y$, $x \& y$ || \neg no



f	no	out
0	0	$x+y$
0	1	$\neg(x+y)$
1	0	$x \& y$
1	1	$\neg(x \& y)$

ALU diagram



zx	nx	zy	ny	f	no	out
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

แนวปฏิบัติ

- ใช้ชิปเซตที่ได้ออกแบบไว้ก่อนหน้านี้
- ชิปที่ทำขึ้นใหม่ ใน Project-2
 - เกิดจากการประกอบกันของชิปใน Project-1
- การสร้างชิปใหม่คือ copy ชิปเก่าแล้วเปลี่ยนชื่อ
 - เติมโค้ด HDL เพียงไม่กี่บรรทัด (เราตั้งใจให้เป็นเช่นนั้น)
- นักศึกษาสามารถผลิตชิป(เขียนโค้ด HDL) เพื่อช่วยงานให้ตนเองทำงานได้ง่าย เราเรียกว่า 'helper chips'
- เพื่อให้การเรียนรู้เป็นอย่างมีประสิทธิภาพ ควรใช้ชิปที่มาจาก การเรียนตามลำดับ แทนการเขียนชิปที่มีคุณสมบัติเกินจำเป็น