



## การจัดองค์การคอมพิวเตอร์

# W1.7 Project Overview

---

31110321 Computer Organization

สำหรับนักศึกษาชั้นปีที่ 2 สาขาวิชาวิศวกรรมคอมพิวเตอร์

ทรงฤทธิ์ กิตีศรีวรพันธุ์

songrit@npu.ac.th

สาขาวิชาวิศวกรรมคอมพิวเตอร์  
มหาวิทยาลัยนครพนม

# Lecture plan

---

- 1.1 Boolean Logic
- 1.2 Boolean Functions Synthesis
- 1.3 Logic Gates
- 1.4 Hardware Description Language
- 1.5 Hardware Simulation
- 1.6 Multi-Bit Buses
- **1.7 Project Overview**

# Project 1

---

- Given : Nand
- Goal : สร้างเกตต่อไปนี้

## Elementary logic gates

- Not
- And
- Or
- Xor
- Mux ←
- DMux ←

## 16-bit variants

- Not16
- And16 ←
- Or16
- Mux16

## Multi-way variants



- Or8Way
- Mux4Way16 ←
- Mux8Way16
- DMux4Way
- DMux8Way

# Project 1

---

- Given : Nand
- Goal : สร้างเกตต่อไปนี้

## Elementary logic gates

- Not
- And
- Or
- Xor
- Mux 
- DMux 

## 16-bit variants

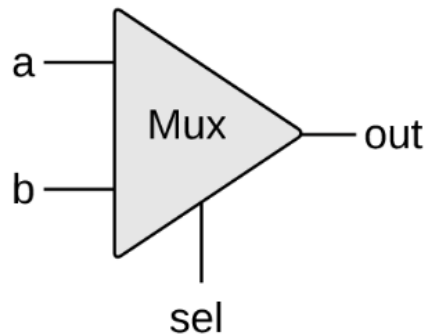
- Not16
- And16
- Or16
- Mux16

## Multi-way variants

- Or8Way
- Mux4Way16
- Mux8Way16
- DMux4Way
- DMux8Way

# Mux

- Multiplexor



```
if (sel==0)
    out=a
else
    out=b
```

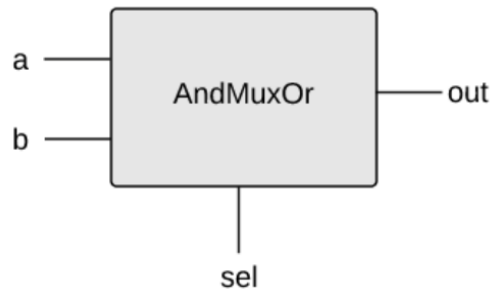
a	b	sel	out
0	0	0	0
0	1	0	0
1	0	0	1
1	1	0	1
0	0	1	0
0	1	1	1
1	0	1	0
1	1	1	1

sel	out
0	a
1	b

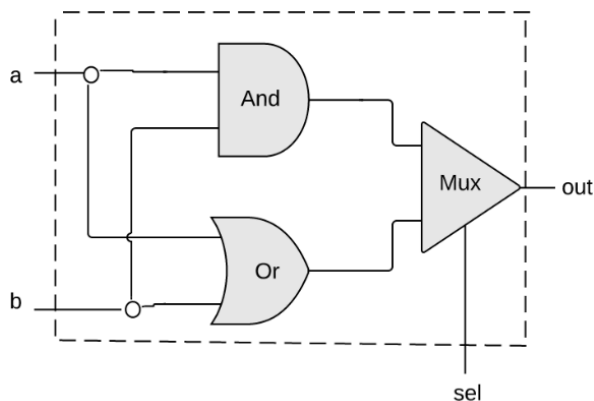
ความหมายเดียวกับ  
truth table

- 2-way multiplexor ใช้ sel เลือกข้อมูลทางใดทางหนึ่ง
- ใช้แพร่หลาย:
  - วงจรดิจิทัล
  - การสื่อสารเครือข่ายคอมพิวเตอร์

# สร้างเกตที่โปรแกรมได้ด้วย mux



```
if (sel==0)
    out = (a And b)
else
    out = (a Or b)
```



a	b	sel	out
---	---	-----	-----

0	0	0	0
0	1	0	0
1	0	0	0
1	1	0	1
0	0	1	0
0	1	1	1
1	0	1	1
1	1	1	1

When sel==0  
the gate acts like  
an And gate

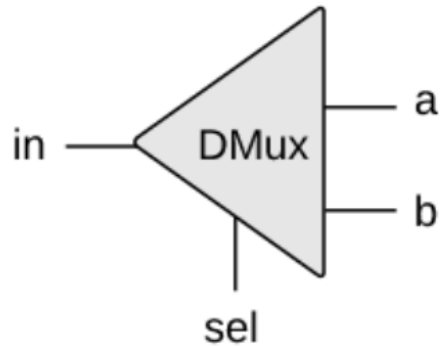
When sel==1  
the gate acts like  
an Or gate

```
CHIP AndMuxOr {
    IN a, b, sel;
    OUT out;

    PARTS:
        And (a=a, b=b, out=andOut);
        Or  (a=a, b=b, out=orOut);
        Mux (a=andOut, b=orOut, sel=sel, out=out);
}
```

# Demux

- Demultiplexor



```
if (sel==0)
    {a,b}={in,0}
else
    {a,b}={0,in}
```

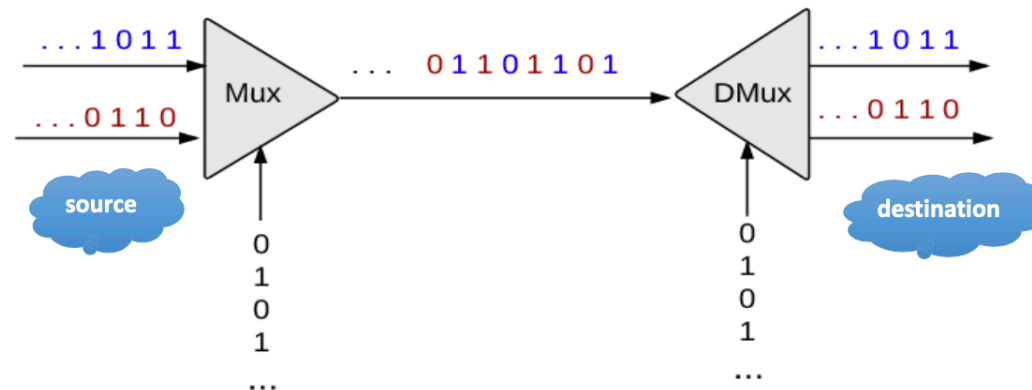
in	sel	a	b
0	0	0	0
0	1	0	0
1	0	1	0
1	1	0	1

- ทำตรงข้าม Mux
- เข้า 1 ออก 2

DMux.hdl

```
CHIP DMux {
    IN in, sel;
    OUT a, b;

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



- มีข้อจำกัดมีสายเส้นเดียว มีข้อมูล 2 ชุด
- เลือกข้อมูลใดข้อมูลหนึ่ง ในหนึ่งเวลา
- sel ต่อกับสวิทช์ใช้เลือกข้อมูล
- sel = 1 เลือก a , sel = 0 เลือก b



# Project 1

---

## Elementary logic gates

- Not
- And
- Or
- Xor
- Mux
- DMux

## 16-bit variants

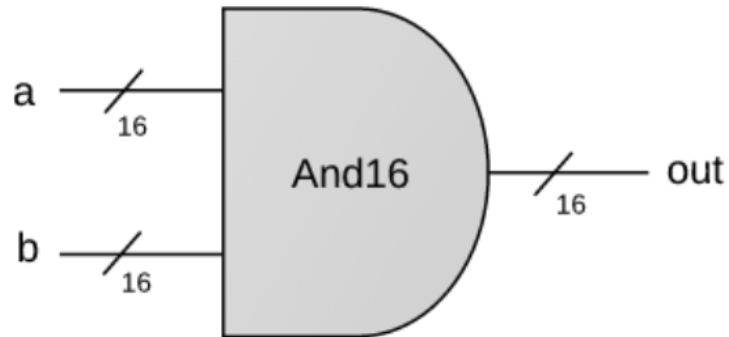
- Not16
- And16
- Or16
- Mux16



## Multi-way variants

- Or8Way
- Mux4Way16
- Mux8Way16
- DMux4Way
- DMux8Way

# And16



a = 1 0 1 0 1 0 1 1 0 1 0 1 1 1 0 0

b = 0 0 1 0 1 1 0 1 0 0 1 0 1 0 1 0

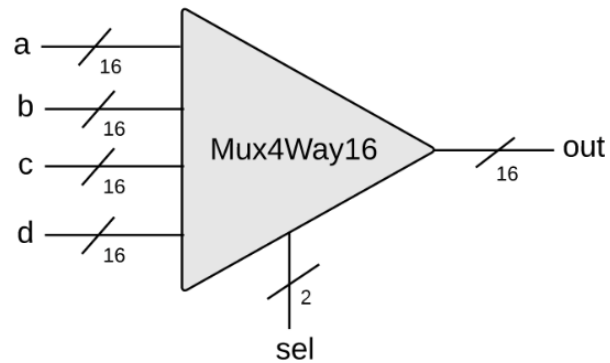
out = 0 0 1 0 1 0 0 1 0 0 0 0 1 0 0 0

```
CHIP And16 {  
    IN a[16], b[16];  
    OUT out[16];  
  
    PARTS:  
    // Put your code here:  
}
```

- ประมวลผลครั้งละ 16 bit
- (ใช้ multi buses)

# 16-bit, 4-way multiplexor

- Mux4Way16



sel[1] ]	sel[0]	out
0	0	<b>a</b>
0	1	<b>b</b>
1	0	<b>c</b>
1	1	<b>d</b>

- คำใบ้

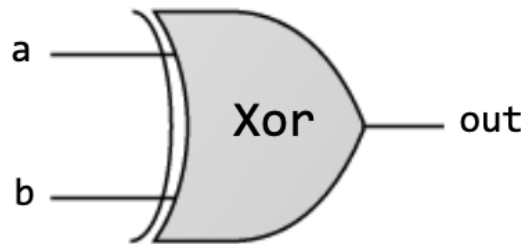
- ใช้ Mux16 หลายตัว

Mux4Way16.hdl

```
CHIP Mux4Way16 {  
  IN a[16], b[16], c[16], d[16],  
      sel[2];  
  OUT out[16];  
  
  PARTS:  
    // Put your code here:  
}
```

# Demo : Chip building

- Xor



outputs 1 if  $a \neq b$

Xor.cmp

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

เมื่อรันโปรแกรม Xor.hdl  
ในไทด์เร็กทอรีที่มีไฟล์  
Xor.tst และ Xor.cmp  
โปรแกรม Hardware  
Simulator จะทดสอบการ  
ทำงานให้อัตโนมัติ

Xor.hdl

```
CHIP Xor {  
    IN  a, b;  
    OUT out;  
  
    PARTS:  
    // Put your code here.  
}
```

Xor.tst

```
load Xor.hdl,  
output-file Xor.out,  
compare-to Xor.cmp,  
output-list a b out;  
set a 0, set b 0, eval, output;  
set a 0, set b 1, eval, output;  
set a 1, set b 0, eval, output;  
set a 1, set b 1, eval, output;
```