課題 1

祝心磊

10/16 2025

1 HDL

1.1 Not

```
CHIP Not {
    IN a,b;
    OUT out;
    PARTS:
    Nand(a=a,b=a,out=out);
}
```

1.2 And

```
CHIP And {
    IN a, b;
    OUT out;
    PARTS:
    Nand(a=b, b=a, out=w1);
    Not(in=w1, out=out);
}
```

1.3 Or

```
CHIP Or {
    IN a, b;
    OUT out;
    PARTS:
    Not(in=a,out=nota);
    Not(in=b,out=notb);
    And(a=nota,b=notb,out=nand);
    Not(in=nand,out=out);
}

```

1.4 Xor

```
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);
}
```

1.5 Multiplexor

```
CHIP Mux {
    IN a, b, sel;
    OUT out;
    PARTS:
    Not(in=sel,out=notsel);
    And(a=notsel,b=a,out=w1);
    And(a=sel,b=b,out=w2);
    Or(a=w1,b=w2,out=out);
}
```

1.6 Demultiplexor

```
CHIP DMux {
    IN in, sel;
    OUT a, b;
    PARTS:
     Not(in=sel,out=notsel);
     And(a=notsel,b=in,out=a);
     And(a=sel,b=in,out=b);
}
```

1.7 複数ビット

1.7.1 Not16

```
CHIP Not16 {
    IN in[16];
    OUT out[16];
    PARTS:
    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]);
}
```

1.7.2 And16

```
CHIP And16 {
    IN a[16], b[16];
    OUT out[16];


    PARTS:
    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]);
}
```

1.7.3 Or

```
CHIP Or16 {
    IN a[16], b[16];
    OUT out[16];

    PARTS:
    Or(a=a[0], b=b[0], out=out[0]);
    Or(a=a[1], b=b[1], out=out[1]);
    Or(a=a[2], b=b[2], out=out[2]);
    Or(a=a[3], b=b[3], out=out[3]);
```

```
    Or(a=a[4], b=b[4], out=out[4]);
    Or(a=a[5], b=b[5], out=out[5]);
    Or(a=a[6], b=b[6], out=out[6]);
    Or(a=a[7], b=b[7], out=out[7]);
    Or(a=a[8], b=b[8], out=out[8]);
    Or(a=a[9], b=b[9], out=out[9]);
    Or(a=a[10], b=b[10], out=out[10]);
    Or(a=a[11], b=b[11], out=out[11]);
    Or(a=a[12], b=b[12], out=out[12]);
    Or(a=a[13], b=b[13], out=out[13]);
    Or(a=a[14], b=b[14], out=out[14]);
    Or(a=a[15], b=b[15], out=out[15]);
}
```

1.7.4 Xor

```
CHIP Xor16 {
    IN a[16], b[16];
    OUT out[16];
    PARTS:
    Xor(a=a[0], b=b[0], out=out[0]);
    Xor(a=a[1], b=b[1], out=out[1]);
    Xor(a=a[2], b=b[2], out=out[2]);
    Xor(a=a[3], b=b[3], out=out[3]);
    Xor(a=a[4], b=b[4], out=out[4]);
    Xor(a=a[5], b=b[5], out=out[5]);
    Xor(a=a[6], b=b[6], out=out[6]);
    Xor(a=a[7], b=b[7], out=out[7]);
    Xor(a=a[8], b=b[8], out=out[8]);
    Xor(a=a[9], b=b[9], out=out[9]);
    Xor(a=a[10], b=b[10], out=out[10]);
    Xor(a=a[11], b=b[11], out=out[11]);
    Xor(a=a[12], b=b[12], out=out[12]);
    Xor(a=a[13], b=b[13], out=out[13]);
    Xor(a=a[14], b=b[14], out=out[14]);
    Xor(a=a[15], b=b[15], out=out[15]);
}
```

1.7.5 Mux16

```
CHIP Mux16 {
    IN a[16], b[16], sel;
    OUT out[16];

    PARTS:
    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]);
}
```

2 HDL の実装

2.1 Xor(排他的論理和回路)

入力 a と b が異なるときに出力 out が 1 となる.
out=(a ∧¬ b) ∨ (¬ a ∧ b) というブール式を HDL で表現している。

2.2 Mux(マルチプレクサ)

入力 *sel* に応じて入力 a または b のどちらかを出力する。
out=(a ∧¬ sel) ∨ (b ∧ sel) というブール式を HDL で表現している。

2.3 DMux(デマルチプレクサ)

入力 *sel* と *in* に応じて,a と b が *in* または 0 を出力する.
(a ∧¬ sel) ∨ (b ∧ sel) というブール式を HDL で表現している.

3 おまけ

ド・アモブルの法則より、

$$Nand(x,y) = Not(And(x,y)) = And(Not(x), Not(y))$$

$$Nand(x,x) = Not(x) \tag{1}$$
$$Nand((Nand(x,x), Nand(y,y)) = Or(x,y) \tag{2}$$
$$Nand(Nand(x,y), Nand(x,y)) = And(x,y) \tag{3}$$

と示せるので、つまり And, Or, Not は全て $Nand$ によって表されるので、
$Xor(x,y) = Or(And(x, Not(y)), And(Not(x), y))$ となるので、前で示したことを用いて

$$\begin{aligned} Xor(x,y) &= Or(And(x, Not(y)), And(Not(x), y)) \\ &= Nand((And(x, Not(y)), And(Not(x), y)) \\ &= Nand((Nand(Nand(x, Nand(y,y))), (Nand(Nand(y, Nand(x,x))))) \end{aligned}$$

が、4 つの $nand$ で実装できないと書きながら気付いた。。。
ここで $(Nand(Nand(x, Nand(y, y)))$ を新たなゲート $Nandthrid(x, y)$ を定義し、
2 回呼び出せば、4 つの $nand$ で Xor を実装された。

$$Xor(x, y) = Nand(Nandthrid(x, y), Nandthrid(y, x))$$