東北大学 工学部 機械知能・航空工学科
2020年度 クラス C D

情報科学基礎 I

4. 論理値と論理演算

(教科書2.1節，2.4節)

大学院情報科学研究科
鏡 慎吾

論理値と論理演算

論理値と論理演算

論理値（真理値）

- 「3 は 4 より小さい」は真 (true): 数値 1 で表す
- 「カエルは哺乳動物である」は偽 (false): 数値 0 で表す

論理演算:

- 論理積 (AND): $A \cdot B$ あるいは単に AB

  他の記法: $A \wedge B,\ A \,\&\, B$

- 論理和 (OR): $A + B$

  他の記法: $A \vee B,\ A \mid B$

- 論理否定 (NOT): $\overline{A}$

  他の記法: $\neg A,\ A',\ !A,\ \tilde{}A$

真理値表とゲート記号

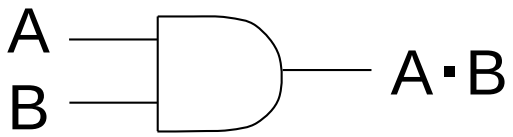
真理値表

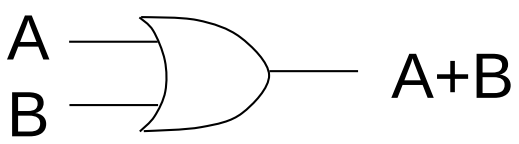
A	B	A･B
0	0	0
0	1	0
1	0	0
1	1	1

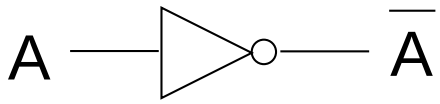
A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

A	$\overline{A}$
0	1
1	0

ゲート記号



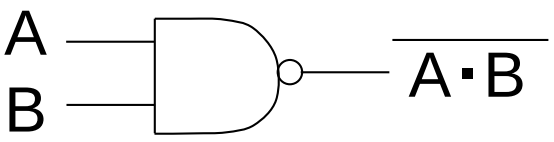




よく使う他の論理演算

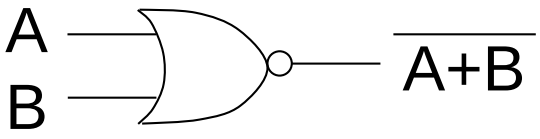
否定論理積 (NAND)

A	B	$\overline{A \cdot B}$
0	0	1
0	1	1
1	0	1
1	1	0



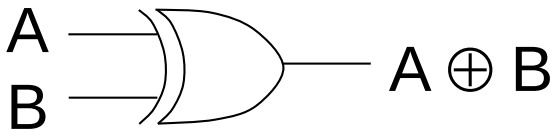
否定論理和 (NOR)

A	B	$\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0



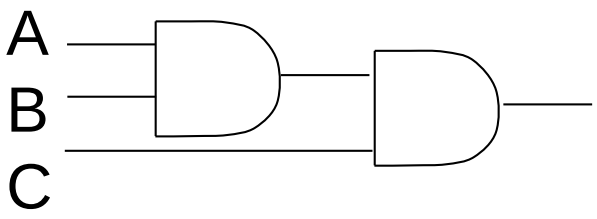
排他的論理和 (eXclusive OR) (XOR)

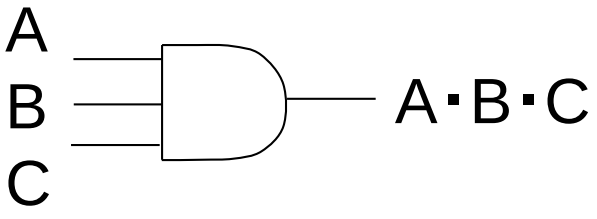
A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

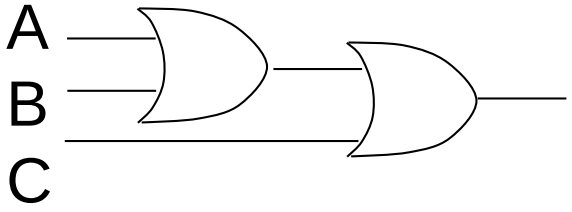


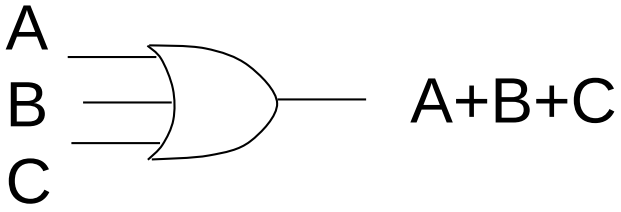
- 片方だけ 1 のときに限って 1
- つまり入力が相異なるときに1

多入力AND, 多入力OR









- 論理積も論理和も，交換則 (AB = BA, A+B = B+A) と結合則 ((AB)C = A(BC), (A+B)+C = A+(B+C)) が成り立つ.
- 入力の順序に関係なく,
  - 多入力ANDは入力が一つでも 0 なら 0
  - 多入力 OR は入力が一つでも 1 なら 1
- 他入力 NAND，他入力NORは，それらの出力を反転したもの

多入力XOR

- 2入力 XOR は，入力のうち 1 の数が奇数なら 1，偶数なら 0
- 多入力 XOR の動作を，入力を 1 つずつ増やしながら考えてみる

- $X \oplus 1 = \overline{X}$; $X \oplus 0 = X$
- A, B, C, ... と順に見ていって，入力に 1 が現れる度に出力は反転する
- 結局，多入力 XOR は，入力のうち 1 の数が奇数なら 1 に，偶数なら 0 になる

例題

(1) 任意の論理値 A に対して以下の条件が満たされるような
論理演算子 ★ を求めよ

A ★ 0 = A
A ★ 1 = 1

真理値表で考えるとよい

A	x	?
0	0	
0	1	
1	0	
1	1	

(2) 同じことを以下の条件について答えよ

A ★ 0 = 0
A ★ 1 = A

(3) 同じことを以下の条件について答えよ

A ★ 0 = A
A ★ 1 = $\overline{A}$

解答例

(1)

A	x	?
0	0	0
0	1	1
1	0	1
1	1	1

OR

(2)

A	x	?
0	0	0
0	1	0
1	0	0
1	1	1

AND

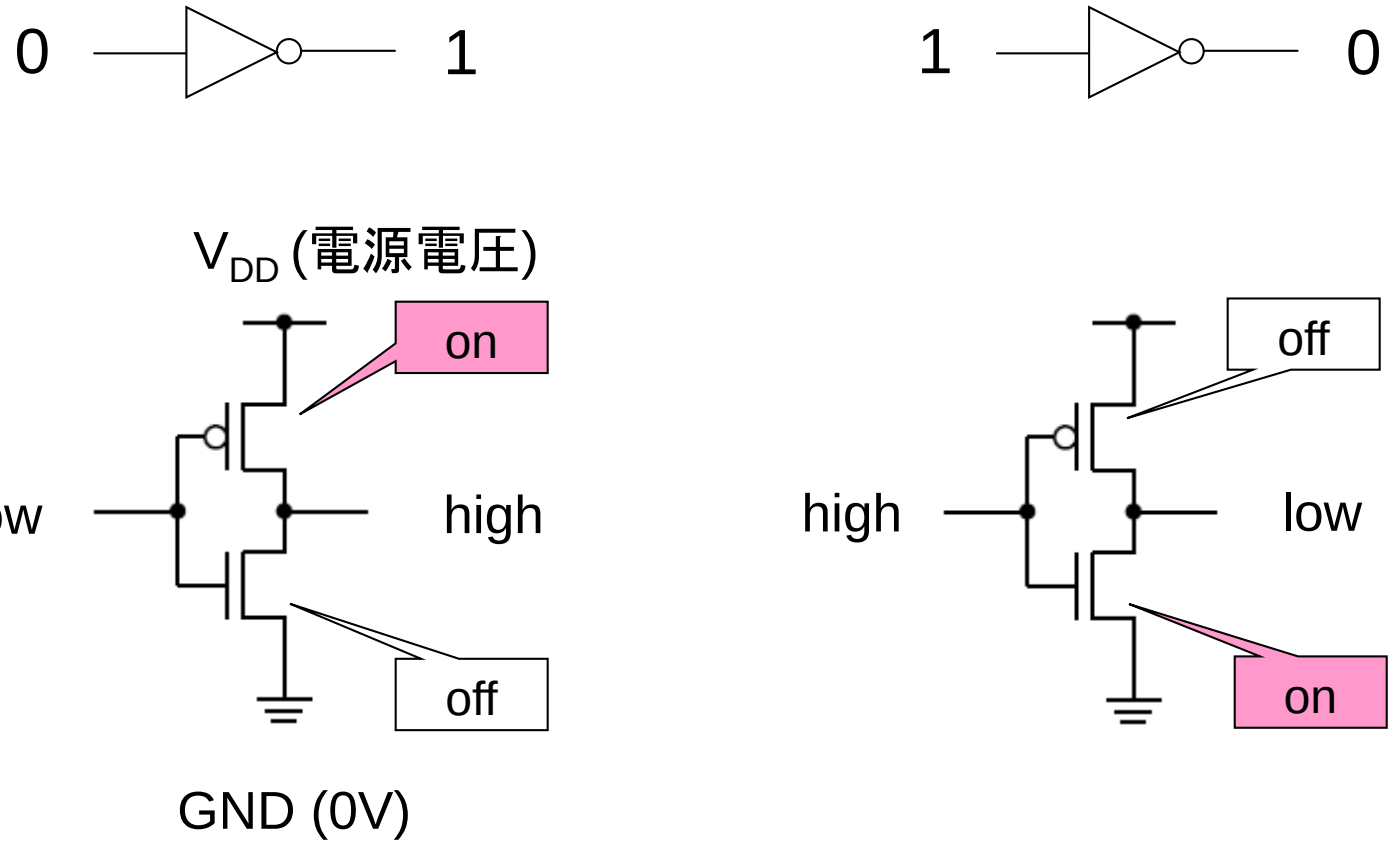
(3)

A	x	?
0	0	0
0	1	1
1	0	1
1	1	0

XOR

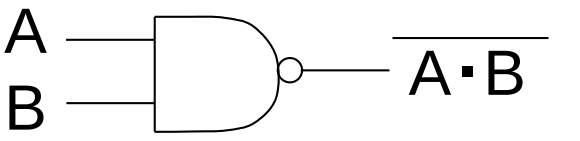
電子回路としての実現

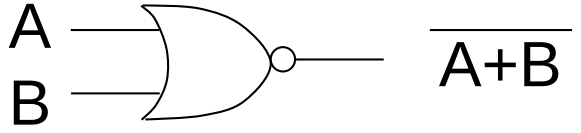
NOTゲート の回路 (インバータ)

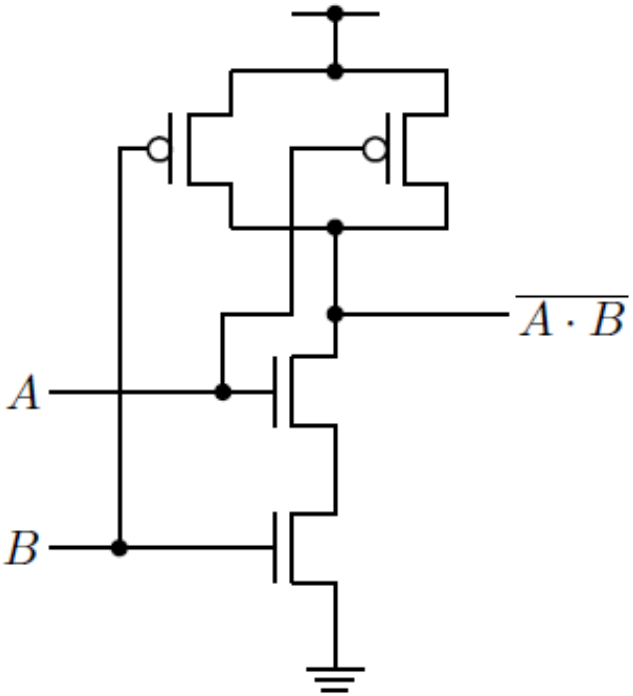


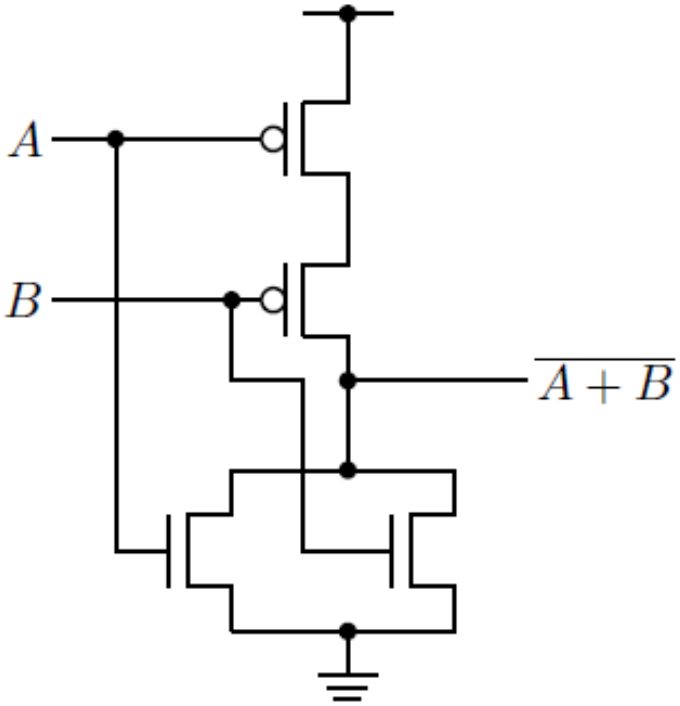
CMOS技術 (Complementary MOS):
NMOS と PMOS を常に対にして使う (低消費電力)

NAND, NORゲートの回路









- AND, OR は NAND, NOR, NOT から作れる
- XOR は AND, OR, NOT から作れる

プログラミングで使う論理演算

論理演算子

```
if (x > 20 && x < 80) {
    ...
}

if (a < 0 || b < 0) {
    ...
}

if (!isalpha(c)) {
    ...
}
```

C言語では，比較演算子 (>, >=, <, <=, ==, !=) は真のとき整数 1 を，偽のとき整数 0 を返す．

演算子 &&, ||, ！ が，論理積，論理和，論理否定を行う．

if や while などの条件部は，0 を偽，0 以外を真とみなして判定する．

ビットごと論理演算子

C言語の演算子 &, |, ~,^ などは，&&, ||, ! とは異なり**ビットごとの**論理演算を行う

```
a = 0x1234;    // 0001 0010 0011 0100
b = 0xcafe;    // 1100 1010 1111 1110
c = a & b;     // 0000 0010 0011 0100 (AND)
c = a | b;     // 1101 1010 1111 1110 (OR)
c = ~a;        // 1110 1101 1100 1011 (NOT)
c = a ^ b;     // 1101 1000 1100 1010 (XOR)
```

任意のビットは

- 1 との OR を取ると 1になる; 0 との OR は元のまま不変
- 0 との AND を取ると 0 になる; 1 との AND は元のまま不変
- 1 との XOR を取ると反転する; 0 との XOR は元のまま不変

などを利用して，ビットごとの操作ができる

例題

ファミリーコンピュータ用ゲーム「ドラゴンクエストIV 導かれし者たち」((株) エニックス, 1990年) では，敵との戦闘中に「にげる」操作を8 回行うと，それ以降，敵へのすべての攻撃が強力なものとなる(会心の一撃と呼ばれる) 現象が生じた．内部でどのような処理が行われていたか推測して述べよ．

(2010年度 期末試験)

http://www.youtube.com/watch?v=_Ao1ofpI0mE

解答例

ドラゴンクエストIV (エニックス) の戦闘状態を保持するメモリ領域の一部は，以下のような構成だったと推測されている:

霧フラグ	?	?	?	会心フラグ	「にげる」コマンドカウンタ		
7	6	5	4	3	2	1	0

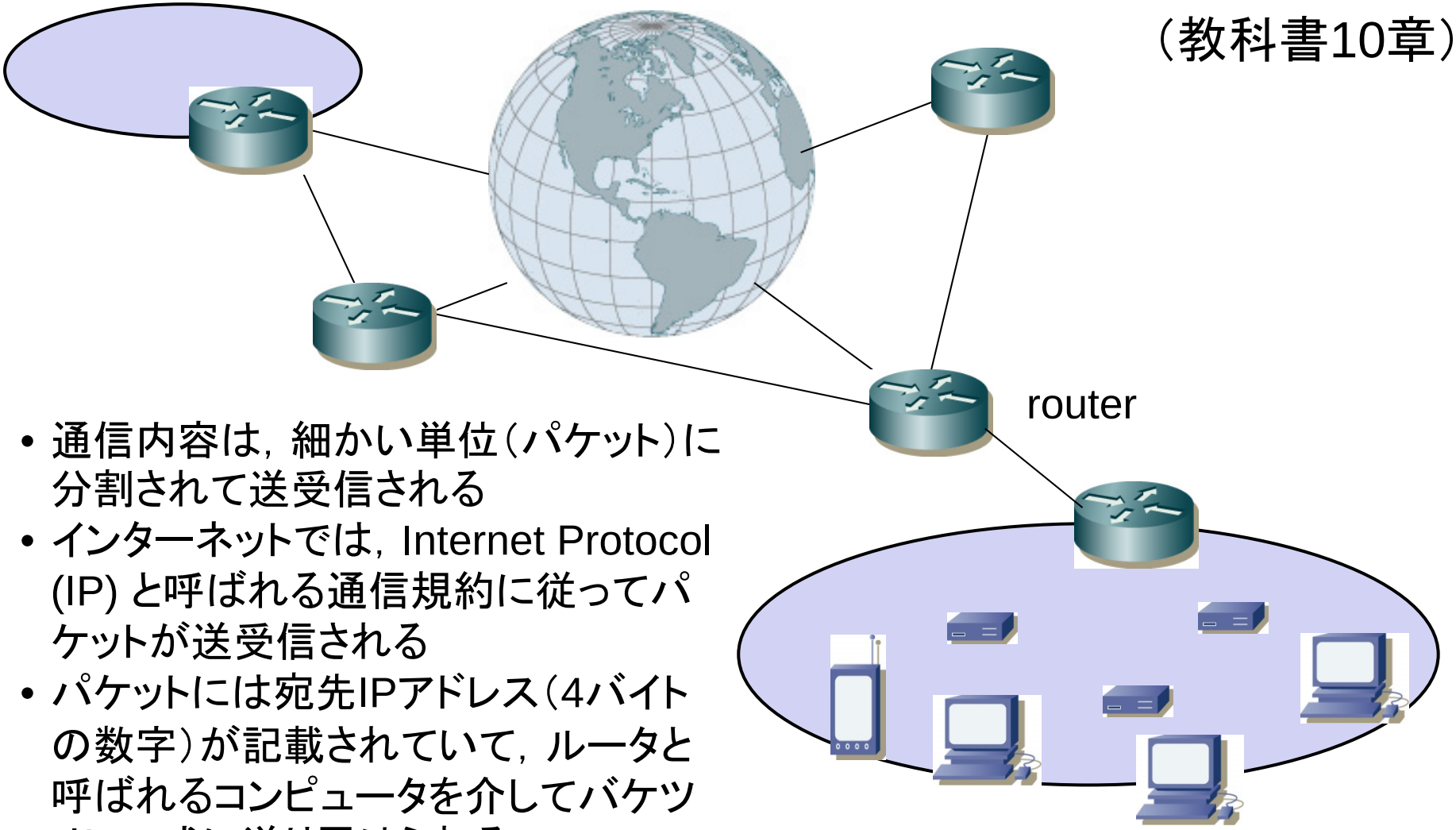
```
critical_hit_bit = 1 << 3;      // 0000 1000
fog_bit = 1 << 7;               // 1000 0000

status = status | critical_hit_bit;  // 会心の一撃ON
status = status & ~critical_hit_bit; // 会心の一撃OFF
status = status ^ fog_bit;           // 霧状態を反転

if (status & fog_bit) {
    // 霧状態ならこの処理を実行
}
```

例: コンピュータネットワーク

（教科書10章）



- 通信内容は，細かい単位（パケット）に分割されて送受信される
- インターネットでは，Internet Protocol (IP) と呼ばれる通信規約に従ってパケットが送受信される
- パケットには宛先IPアドレス（4バイトの数字）が記載されていて，ルータと呼ばれるコンピュータを介してバケツリレー式に送り届けられる

Windows 10 の コントロールパネル → ネットワークとインターネット
→ 状態 → ネットワークのプロパティを表示

IPv4 アドレス:	10.240.10.191/24
IPv6 アドレス:	fe80::d461:31eb:2998
デフォルト ゲートウェイ:	10.240.10.1

「/24」はサブネットマスク長

（自分の）アドレス	00001010	11110000	00001010	10111111
サブネットマスク	11111111	11111111	11111111	00000000

- 自分のアドレスとサブネットマスクのビットごとANDと，宛先アドレスとサブネットマスクのビットごとANDを比較する．等しければ同じLANの中にいるので，LAN内で定められた手順によって通信する
- 等しくなければ，デフォルトゲートウェイ（デフォルトルータ）として設定されているルータへ送信して，あとは任せる

デフォルトゲートウェイ	00001010	11110000	00001010	00000001

練習問題

1. 16ビットの値 x が与えられたとき，上位 12 ビットと下位 4 ビットを入れ替えた結果を変数 y に得る処理を，シフト演算とビットごと論理演算の組合せで実現せよ．（このような処理は「4 ビット右ローテート」と呼ばれる）

2. 16ビットの値が与えられたとき，下位から 7 ビット目（ただし LSB を1 ビット目と数えることにする）の値を変数 y に得る処理を，（if 文などを使わずに）シフト演算とビットごと論理演算の組合せで実現せよ．

解答例

1. C言語であれば，以下のような処理で実現できる．一通りとは限らない

```
unsigned short x, y;
...

y = (x << 12) | (x >> 4);
```

2.
```
y = (x >> 6) & 1;
```

あるいは

```
y = (x & (1 << 6)) >> 6;
```