

演習問題

(1) $\pm 87, \pm 50, \pm 26$ を、
8 bit の “2 の補数表示” で表せ。

(2) 上の表示を用いて、次を筆算で計算せよ。

- 9bit 目への繰上がりが発生するものは？
- 桁溢れが発生しているものは？

(1) $(+87) + (+26)$ (2) $(+87) + (-26)$

(3) $(-87) + (+26)$ (4) $(-87) + (-26)$

(5) $(+87) + (+50)$ (6) $(+87) + (-50)$

(7) $(-87) + (+50)$ (8) $(-87) + (-50)$

(3) 桁溢れの発生を判定するには？

前回の演習問題への注

- 9bit 目への繰上がり \neq 桁溢れ
- 桁溢れせず $\iff -2^7 \leq \text{計算結果} \leq 2^7 - 1$
ではあるが、
 - ★ 演算結果を読み取ったら常にこの範囲
 - ★ 別途計算する訳にはいかない \longrightarrow どうやって判定するのか、という問題
- 桁溢れ判定を演算回路として実現するには、
どの bit の情報を見れば良いか、
まで具体的に考える必要がある

論理回路

- **組合せ回路** :
入力 (の組) によって出力が決まる
→ **演算**に用いる

- **順序回路** :
内部状態を保持し、
入力と入力前の状態とによって
出力と出力後の状態とが決まる
→ **データ (bit 情報) の保持**に用いる

論理回路の基本部品 : 論理素子 (論理ゲート)

- NOT: 否定 : $\neg A$
- OR: 論理和 : $A \vee B$
- AND: 論理積 : $A \wedge B$
- XOR: 排他的論理和 :
 $A \oplus B = (A \wedge \neg B) \vee (\neg A \wedge B)$
- NOR: 論理和の否定 :
 $\neg(A \vee B) = \neg A \wedge \neg B$
- NAND: 論理積の否定 :
 $\neg(A \wedge B) = \neg A \vee \neg B$

論理素子の真理値表

NOT		X
A	0	1
	1	0

OR		B	
		0	1
A	0	0	1
	1	1	1

AND		B	
		0	1
A	0	0	0
	1	0	1

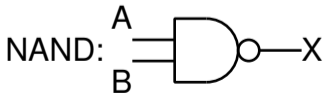
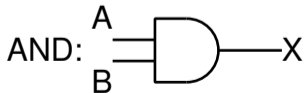
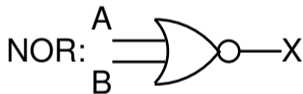
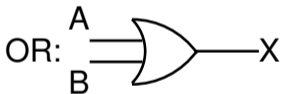
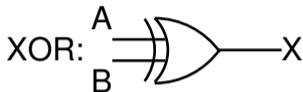
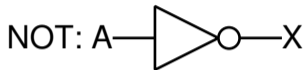
XOR		B	
		0	1
A	0	0	1
	1	1	0

NOR		B	
		0	1
A	0	1	0
	1	0	0

NAND		B	
		0	1
A	0	1	1
	1	1	0

論理素子の MIL 記号

論理素子を回路図で表現するのに用いる記号

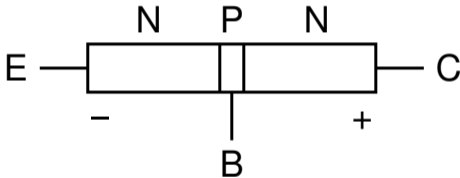


論理素子の物理的な実現

- 電磁石 (リレー)
- 真空管
- 半導体素子 (トランジスタ・IC・LSI)

トランジスタ (NPN 接合型) の仕組み

ベース (B) の電位を制御することで、
エミッタ (E)-コレクタ (C) 間の電流が
大きく変化する



- 情報の増幅
- 情報の伝達 (スイッチング)

トランジスタ (NPN 接合型) の仕組み

- トランジスタ 1 つに適切に配線
→ NOT ゲート
- トランジスタ 2 つを並列に接続
→ OR ゲート
- トランジスタ 2 つを直列に接続
→ AND ゲート

論理素子 (論理ゲート)

- NOT: 否定 : $\neg A$
- OR: 論理和 : $A \vee B$
- AND: 論理積 : $A \wedge B$
- XOR: 排他的論理和 :
 $A \oplus B = (A \wedge \neg B) \vee (\neg A \wedge B)$
- NOR: 論理和の否定 :
 $\neg(A \vee B) = \neg A \wedge \neg B$
- NAND: 論理積の否定 :
 $\neg(A \wedge B) = \neg A \vee \neg B$

論理素子 (論理ゲート)

これらの論理素子を組み合わせ、
基本的な演算を実現しよう

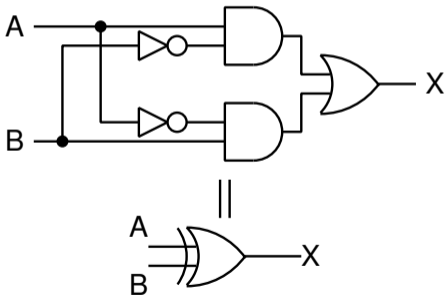
という訳だが、

実はこれらの論理素子の一部のみを用いて
他の論理素子が表せる

論理素子

実はこれらの論理素子の一部のみを用いて
他の論理素子が表せる

例：XORをNOT, OR, ANDで表す



論理素子

どんな論理回路も

これらの論理素子の組合せで表せるか？

→ “論理回路・論理素子が表すもの”
の定式化が必要

→ “Boole 関数” (真理値 (の組) を値とする関数)

論理素子

どんな論理回路も

これらの論理素子の組合せで表せるか？

→ “論理回路・論理素子が表すもの”

の定式化が必要

→ “Boole 関数” (真理値 (の組) を値とする関数)

論理素子

どんな論理回路も

これらの論理素子の組合せで表せるか？

→ “論理回路・論理素子が表すもの”
の定式化が必要

→ “**Boole 関数**” (真理値 (の組) を値とする関数)

組合せ回路

入力 (の組) によって出力が決まる

n 入力 m 出力の回路は **Boole** 関数

$$f : X_f \longrightarrow \{0, 1\}^m$$

(ここに $X_f \subset \{0, 1\}^n$ は許される入力全体)
を定める

組合せ回路 : **Boole** 関数の論理素子による実現

組合せ回路

NOT, OR, AND のみで、

全ての **Boole** 関数可以实现できる

$$f(A_1, \dots, A_n) = (X_{11} \wedge \dots \wedge X_{1t_1})$$

$$\vee \dots$$

$$\vee (X_{s1} \wedge \dots \wedge X_{st_s})$$

(各 X_{ij} は A_k または $\neg A_k$)

... 論理和標準形・選言標準形

(disjunctive normal form, DNF)

組合せ回路

双対的に、次の形でも書ける：

$$\begin{aligned} f(A_1, \dots, A_n) = & (X_{11} \vee \dots \vee X_{1t_1}) \\ & \wedge \dots \\ & \wedge (X_{s1} \vee \dots \vee X_{st_s}) \end{aligned}$$

(各 X_{ij} は A_k または $\neg A_k$)

… 論理積標準形・連言標準形
(conjunctive normal form, CNF)

組合せ回路

- NOT, OR, AND のみで、
 全ての Boole 関数が実現できる
- NOT があれば OR, AND は片方で良い
- OR, AND だけでは
 全ての Boole 関数は実現できない
- NOR または NAND 一種類だけで、
 全ての Boole 関数が実現できる

以下では、NOT, OR, AND を用いた実現を考える

組合せ回路

- NOT, OR, AND のみで、
 全ての Boole 関数が実現できる
- NOT があれば OR, AND は片方で良い
- OR, AND だけでは
 全ての Boole 関数は実現できない
- NOR または NAND 一種類だけで、
 全ての Boole 関数が実現できる

以下では、NOT, OR, AND を用いた実現を考える

組合せ回路

- NOT, OR, AND のみで、
 全ての Boole 関数が実現できる
- NOT があれば OR, AND は片方で良い
- OR, AND だけでは
 全ての Boole 関数は実現できない
- NOR または NAND 一種類だけで、
 全ての Boole 関数が実現できる

以下では、NOT, OR, AND を用いた実現を考える

組合せ回路

- NOT, OR, AND のみで、
 全ての Boole 関数が実現できる
- NOT があれば OR, AND は片方で良い
- OR, AND だけでは
 全ての Boole 関数は実現できない
- NOR または NAND 一種類だけで、
 全ての Boole 関数が実現できる

以下では、NOT, OR, AND を用いた実現を考える

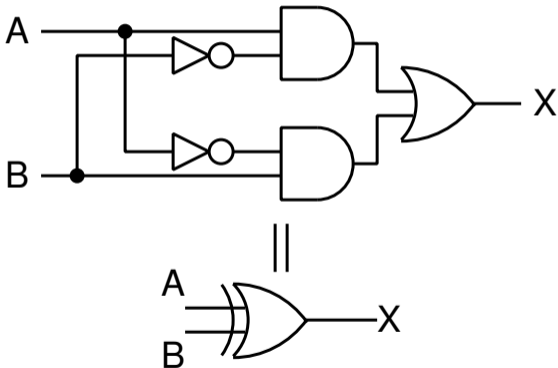
組合せ回路

- NOT, OR, AND のみで、
 全ての Boole 関数が実現できる
- NOT があれば OR, AND は片方で良い
- OR, AND だけでは
 全ての Boole 関数は実現できない
- NOR または NAND 一種類だけで、
 全ての Boole 関数が実現できる

以下では、NOT, OR, AND を用いた実現を考える

組合せ回路

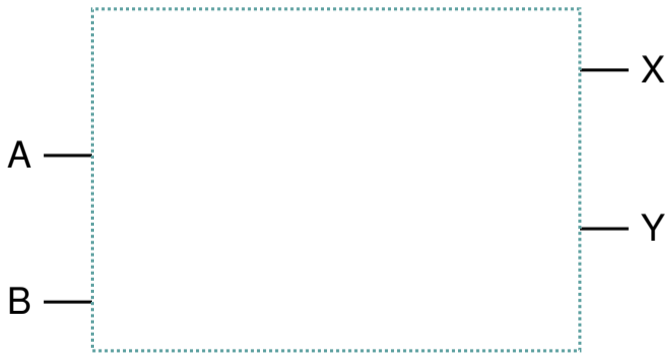
例：XOR を NOT, OR, AND で表す



半加算器 (semi adder, SA)

入力 : **A, B**: 各桁の値

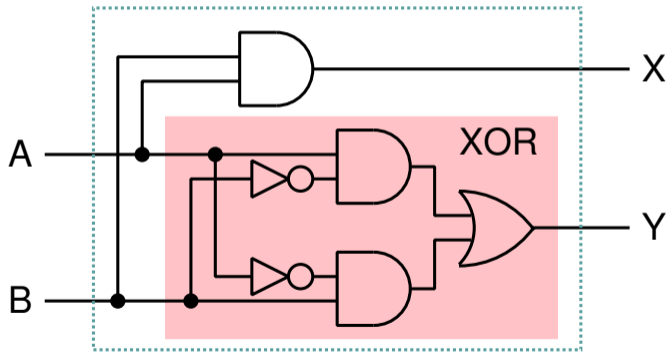
出力 : **X**: 上への繰上がり, **Y**: 当桁の値



半加算器 (semi adder, SA)

入力 : **A, B**: 各桁の値

出力 : **X**: 上への繰上がり, **Y**: 当桁の値



全加算器 (full adder, FA)

1桁の加算は半加算器で出来るが、

2桁以上の加算の場合は、

下の桁からの繰上がりにも対応した回路が必要

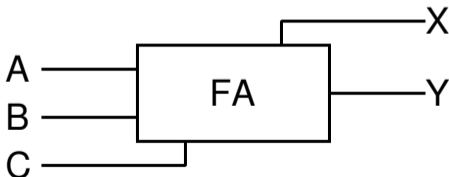
- 入力 : **A, B**: 各桁の値, **C**: 下からの繰上がり
- 出力 : **X**: 上への繰上がり, **Y**: 当桁の値

→ **全加算器 (full adder, FA)**

演習問題

全加算器 (full adder, FA) を、
NOT, OR, AND を用いて構成せよ

- 入力 : **A, B**: 各桁の値, **C**: 下からの繰上がり
- 出力 : **X**: 上への繰上がり, **Y**: 当桁の値



演習問題

全加算器 (**full adder**) を部品として用いて、
二進 4 桁の符号付き整数値 2 つの
加算を行なう組合せ回路を構成せよ。
但し、桁溢れの発生を判定し、
桁溢れが生じた場合は **overflow flag** を立てよ。

- 入力 : $A_1, A_2, A_3, A_4; B_1, B_2, B_3, B_4$
 : 数値 (A_1, B_1 は符号 bit)
- 出力 :
 X_1, X_2, X_3, X_4 : 加算の結果 (X_1 は符号 bit)
 Y : **overflow flag**
 (桁溢れが発生したら 1、しなければ 0)

(再掲) 論理回路

- データ (bit 情報) の保持 → 順序回路
- 基本的な演算 → 組合せ回路

組合せ回路

入力 (の組) によって出力が決まる (演算回路)

n 入力 m 出力の回路は **Boole** 関数

$$f : X_f \longrightarrow \{0, 1\}^m$$

(ここに $X_f \subset \{0, 1\}^n$ は許される入力全体)

(再掲) 論理回路

- データ (bit 情報) の保持 → 順序回路
- 基本的な演算 → 組合せ回路

組合せ回路

入力 (の組) によって出力が決まる (演算回路)

n 入力 m 出力の回路は **Boole** 関数

$$f : X_f \longrightarrow \{0, 1\}^m$$

(ここに $X_f \subset \{0, 1\}^n$ は許される入力全体)

順序回路

- 内部状態を保持し、
- 入力と入力前の状態とによって、
- 出力と出力後の状態が決まる

許される内部状態: $Q_f \subset \{0, 1\}^B$

n 入力 m 出力の順序回路は **Boole** 関数

$$f : Q_f \times X_f \longrightarrow Q_f \times \{0, 1\}^m$$

(ここに $X_f \subset \{0, 1\}^n$ は許される入力全体)
を定める

順序回路

- 内部状態を保持し、
- 入力と入力前の状態とによって、
- 出力と出力後の状態が決まる

許される内部状態: $Q_f \subset \{0, 1\}^B$

n 入力 m 出力の順序回路は **Boole** 関数

$$f : Q_f \times X_f \longrightarrow Q_f \times \{0, 1\}^m$$

(ここに $X_f \subset \{0, 1\}^n$ は許される入力全体)
を定める

順序回路

内部状態を計算機内に如何に保持するか

- コンデンサに電荷を蓄える
- 複数の安定状態を持つ論理回路で実現
例: フリップフロップ

順序回路

内部状態を計算機内に如何に保持するか

- コンデンサに電荷を蓄える
- 複数の安定状態を持つ論理回路で実現
例: フリップフロップ