仙台市/仙台市産業振興事業団

ロボット博士の基礎からのメカトロニクスセミナー

C03/Rev 1.0

第３回

# デジタルの基礎

仙台市地域連携フェロー

熊　谷　正　朗

kumagai@tjcc.tohoku-gakuin.ac.jp

東北学院大学工学部
ロボット開発工学研究室 RDE

## 今回の目的

○ デジタルの基礎

テーマ１：デジタルの理論と2進数
- 演算/処理の基礎ルール

テーマ２：デジタルの電気信号
- 表現方法と実現方法

テーマ３：デジタル回路の実際
- ロジック回路
- 組み合わせ回路と順序回路
- コンピュータのしくみ



## イントロダクション

○ デジタルとアナログ

アナログ
- 連続的な値（1と1.00……01は異なる）。
- 世の中のほぼ全ての現象はアナログ。

デジタル
- いくつかの明確に区別できる値に限定：
  「0か1か」（※0/1限定ではない）
- 中間を無視することで
  曖昧さの排除 / 強さ



## イントロダクション

○ なぜデジタルか？

信号の劣化が起きにくい

～ 電気信号にはノイズがつきもの

アナログ：元の信号に少しでも変動が出ると、それが表している「値」の誤差になってしまう。

デジタル：信号を受けるとき「大きい」「小さい」などで解釈するため、そこそこの変動まで耐えられる。

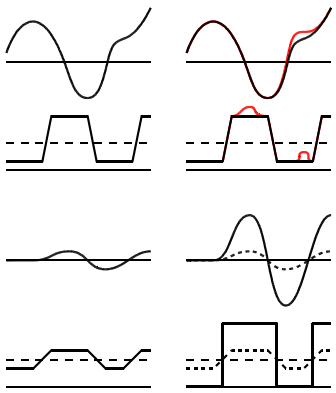
回路素子が単純

アナログ：「比例関係」に非常に気を使う

デジタル：とにかく振り切れれば良い

## イントロダクション

### ○ なぜデジタルか？



アナログでは少しの電圧変化も値の変化として残る。

デジタルでは閾値(しきいち，中間の基準)を超えなければ、値が変わらない。

アナログは「きっちり」倍率の決まった回路が必須。

デジタルは閾値からの差を、「倍率問わず」に、とにかく拡大するだけ。



## イントロダクション

### ○ デジタルの弱点

一本の信号線で同時に表せる情報が少ない

アナログ：現実的には千～十万段階くらい
（回路の作り方に大きく依存）

デジタル：一般に2段階
（数段階にする方法もある）

→ 情報を多くしようとすると

1：線の本数を多くする（平行に並ぶ多数の線）
2：1本の線で高速に切り替えて順に送る（シリアル）



## デジタルの理論と2進数

### ○ 基礎理論：ブール代数

- 「かつ」「もしくは」「ではない」
- 真理値表 と ド・モルガンの定理
- 補助演算子

### ○ 2進数：数を表すルール

- データ表現の基本的発想
- 2進数
- 2進数の演算



## 基本理論：ブール代数

### ○ 2値の理論

状態を2種類のみに限定

- Yes と No のみ、「中くらい」を無視。
- 真/偽、T(rue)/F(alse)、1/0 （、H/L）

基本の演算を3種類のみに限定

- 論理積　AND 「かつ」　[なし][・][×]
- 論理和　OR 「もしくは」　[＋]
- 否　定　NOT 「ではない」 [上に横線][/]

## 基本理論：ブール代数

### ○ 演算のルール

論理積　A AND B、A･B、AB

「 A=1【かつ】B=1 なら A･Bは1 」

A=0, B=0 なら　A･Bは 0
A=0, B=1 なら　A･Bは 0
A=1, B=0 なら　A･Bは 0
A=1, B=1 なら　A･Bは 1

※両方 1 なら 結果が 1



## 基本理論：ブール代数

### ○ 演算のルール

論理和　A OR B、A＋B

「 A=1【もしくは】B=1 なら A＋Bは1 」

A=0, B=0 なら　A＋Bは 0
A=0, B=1 なら　A＋Bは 1
A=1, B=0 なら　A＋Bは 1
A=1, B=1 なら　A＋Bは 1

※どちらか 1 なら 結果が 1



## 基本理論：ブール代数

### ○ 演算のルール

否　定　NOT A、$\overline{A}$、/A

「 A=1【でなければ】$\overline{A}$は1 」

A=0 なら　$\overline{A}$は1
A=1 なら　$\overline{A}$は0

※0と1が逆になる



## 基本理論：真理値表

### ○ 表による演算結果の表現

| A | B | A･B | A＋B | $\overline{A}$ | $\overline{B}$ |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 | 0 |

・入力の全ての組み合わせに対する、
　演算結果を表示する表。
・入力が n本あると2×2…2(n回)＝$2^n$行。
・「状態問わず」で「X」を書くこともある。

## 基本理論：ド・モルガンの法則

○ $\overline{A \cdot B} = \overline{A} + \overline{B}$　　$\overline{A + B} = \overline{A} \cdot \overline{B}$

※[AとB]のANDのあとでNOT　※[AとB]のORのあとでNOT

| A | B | $\overline{A}$ | $\overline{B}$ | A･B | A＋B | $\overline{A \cdot B}$ | $\overline{A}+\overline{B}$ | $\overline{A+B}$ | $\overline{A} \cdot \overline{B}$ |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | | |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | | |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | | |
| 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | | |

等しい

やってみましょう

・真理値表で同じ結果→機能的に等しい

・「 今日は[晴れ]で[暑い] 」 ではないのは
「 今日は[晴れではない] 」 もしくは
「 今日は[暑くない] 」



## 基本理論：補助演算子

○ AND，OR，NOT 以外の演算

| A | B | A NAND B | A NOR B | A XOR B | A XNOR B |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 |

・回路で良くつかわれる演算の定義
※全て、AND，OR，NOT の組み合わせ

・ NAND=Not-AND，NOR=Not-OR（なんど、のあ）
XOR=eXclusiveOR/排他的論理和，XNOR=Not-XOR
XOR：異なると1，XNOR:一致すると1



## 2進数：データ表現の基本的発想

○ 0/1だけで様々な情報を表す

0/1を束にする

・一つの[0か1]では2種類の情報しか
表せない。

・二つの[0か1]では4種類＝2×2の
表現ができる。

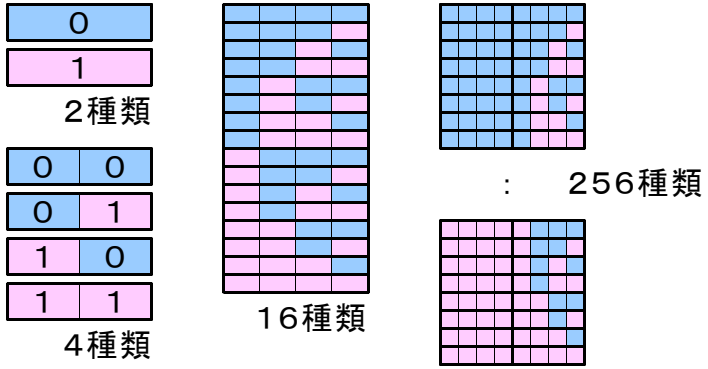
：

・N個の[0か1]で、2のN乗 種類 になる。
2のN乗＝$2^N$＝2×2×…(全部でN個)×2



## 2進数：データ表現の基本的発想

○ 0と1の組み合わせ 1,2,4,8個の場合



## 2進数：データ表現の基本的発想

○ 0/1だけで様々な情報を表す

・N個(Nビット)の[0か1」で、2のN乗 種類
2のN乗＝$2^N$＝2×2×…(全部でN個)×2

この「種類」を使い方に応じて割り当てる
・数値（整数、正負、小数(固定、浮動)）
・文字
・状態（例:OFF/起動中/待機中/動作中）

※ただし、文字、状態などは 0/1⇔整数⇔文字など
と、一度整数を介すことが一般的



## 2進数：データ表現の基本的発想

○ 数値の表現：2進数

ルール：普通の数(十進数)と同じように考える

十：0,1,2,…,8,9, 10,11…98,99, 100,101…

二：0,1, 10,11, 100,101,…111, 1000,…
「2」が出そうになったら、もう上の桁

→N桁で0～$2^N$−1を表せる

そのまま使う→

| | |
|---|---|
| 0 | 0 |
| 0 | 1 |
| 1 | 0 |
| 1 | 1 |



## 2進数：データ表現の基本的発想

○ 数値の表現：2進数、正負、16進数

| 2進 | 正 | ± | 16 | 2進 | 正 | ± | 16 |
|---|---|---|---|---|---|---|---|
| 0000 | 0 | 0 | 0 | 1000 | 8 | −8 | 8 |
| 0001 | 1 | 1 | 1 | 1001 | 9 | −7 | 9 |
| 0010 | 2 | 2 | 2 | 1010 | 10 | −6 | A |
| 0011 | 3 | 3 | 3 | 1011 | 11 | −5 | B |
| 0100 | 4 | 4 | 4 | 1100 | 12 | −4 | C |
| 0101 | 5 | 5 | 5 | 1101 | 13 | −3 | D |
| 0110 | 6 | 6 | 6 | 1110 | 14 | −2 | E |
| 0111 | 7 | 7 | 7 | 1111 | 15 | −1 | F |

2進数で[abcd]→10進数: a×8 ＋ b×4 ＋ c×2 ＋ d
＝ a×$2^3$ ＋ b×$2^2$ ＋ c×$2^1$ ＋ d×$2^0$



## 2進数：データ表現の基本的発想

○ 数値の表現：2進数、正負、16進数

| 2進 | 正 | ± | 16 | 2進 | 正 | ± | 16 |
|---|---|---|---|---|---|---|---|
| 00000000 | 0 | 0 | 00 | 10000000 | 128 | −128 | 80 |
| 00000001 | 1 | 1 | 01 | 10000001 | 129 | −127 | 81 |
| 00000010 | 2 | 2 | 02 | 10000010 | 130 | −126 | 82 |
| : | : | : | : | : | : | : | : |
| : | : | : | : | : | : | : | : |
| 01111101 | 125 | 125 | 7D | 11111101 | 253 | −3 | FD |
| 01111110 | 126 | 126 | 7E | 11111110 | 254 | −2 | FE |
| 01111111 | 127 | 127 | 7F | 11111111 | 255 | −1 | FF |

最上位＝左端が正負↑ ↑−256↑
負の数のほうが1個多い↑

## ２進数：こういうときに必須

### ○ コンピュータ周りは２進数

純ソフトウエア系

- 8bit，16bit，32bit で表せる数値の限界
- 複数の情報をまとめる（例8bit←aabbbccd）
- コンピュータの気持ちでプログラミング

組込ソフトウエア系

- 数値＝２進数→信号の入出力
- ビットごとの処理

※16進数は２進数の代わりに多用される



## ２進数：演算：四則演算

※参考：page19

### ○ 加算（減算）

0＋0＝ 0
0＋1＝ 1
1＋0＝ 1
1＋1＝10

```
  0110      6      6
+ 1101   + 13   + −3
 (1)1
 10011     19      3
```

- 十進数の加算と、[0,1のみ]以外は同じ。
- 「繰り上がり」が発生しやすい。
- 解釈によっては正負の加算(引き算)も可。

※論理和(OR)の＋と混同に注意。
※一番上にあふれた繰り上がりの扱いは要注意。



## ２進数：演算：四則演算

### ○ 乗算 ～「１×１＝１」

3×5＝15
6×7＝42

0×0＝ 0
0×1＝ 0
1×0＝ 0
1×1＝ 1

```
   0011        0110
 × 0101      × 0111
   0011        0110
  0000        0110
 0011        0110
0000        0000
00011111    01010110
```

- 乗算そのものの手間はない。（0か1か）
- 桁ごとの乗算のあとの、加算が面倒。

※負の乗算では符号の処理は別途行う



## ２進数：演算：四則演算

### ○ 除算 ～ 引き算だけできればいい

```
     0110
11 ) 10011
     00
     100
      11
      11
       11
       01
       00
   あまり…1
```

19÷3＝
6あまり1

- 商を立てる判定は「ただの大小チェック」。＝小学生の悩みが不要
- 繰り返しの引き算。

※似た計算方法を使って平方根も簡単に求まる

## 2進数：演算：補助

### ○ ×(−1)，シフト演算

負の数（演算で，減算で）

・「×(−1)」＝「全部NOTして ＋1する」

例 0101（5）→ 1010（−6）→ 1011（−5）

全NOT　＋1

シフト演算（左右にずらす ×$2^n$、÷$2^n$）

・左シフト 0011（3）→ 0110（6） ×2

・右シフト 1000（8）→ 0010（2） ÷4

※一般に0を導入　例外:左端 1000(−8)→1110(−2)

※消える桁がある　例：1011(11)→10(2)（11÷4=2余3）



## 今回の目的

### ○ デジタルの基礎

テーマ1：デジタルの理論と2進数

・演算/処理の基礎ルール

テーマ2：デジタルの電気信号

・表現方法と実現方法

テーマ3：デジタル回路の実際

・ロジック回路

・組み合わせ回路と順序回路

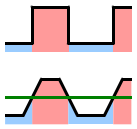
・コンピュータのしくみ



## デジタルの電気信号

### ○ デジタルの回路における表現

一般的なデジタル信号

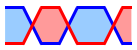


・1本の線で一つのデジタル値を表す。

（a）電圧の高低（低=0，高=1，または逆）

（b）電流の有無/大小（有,大=1など）

差動デジタル信号（高速通信向き）



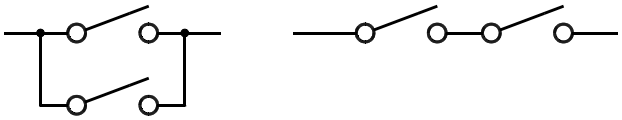
・2本の信号線を対で使い，相対的な電圧の高低(A>BとA<B)で一つの値。

・耐ノイズ性が高い（詳しくは次回）



## デジタル回路の実現

### ○ スイッチによる実現



スイッチの並列つなぎ

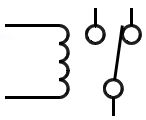
→ どちらかONなら電気が流れる ＝ OR

スイッチの直列つなぎ

→ ともにONなら電気が流れる ＝ AND

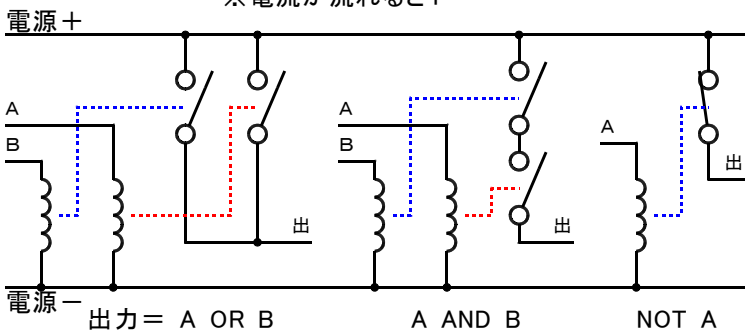
## デジタル回路の実現

○ リレーによる実現



リレー：
電磁石でON/OFF，切替

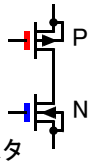
※電流が流れると1






## デジタル回路の実現

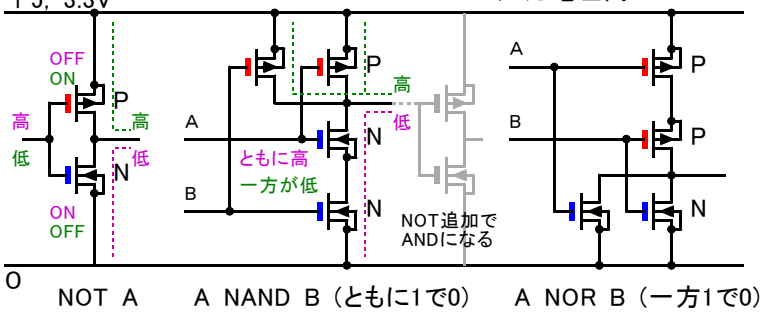
○ MOSトランジスタで実現




CMOS型回路

MOSトランジスタ
P:入力電圧低→ON
N:入力電圧高→ON

※電圧が高いと1






## デジタル回路の実現

○ ブール代数が実現できればOK

AND OR NOT があればいい

演算回路の実現

昔> 真空管，リレー →

トランジスタ →

DTL，TTL（ダイオード＆トランジスタ）→

CMOS（相補型 MOS FET）＜今

より速く、より小さく、より省電力



## 今回の目的

○ デジタルの基礎

テーマ1：デジタルの理論と2進数
- 演算/処理の基礎ルール

テーマ2：デジタルの電気信号
- 表現方法と実現方法

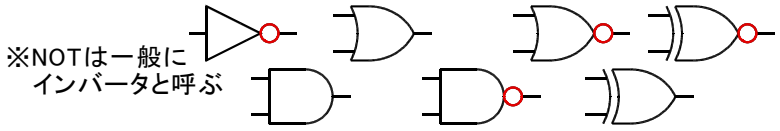
テーマ3：デジタル回路の実際
- ロジック回路
- 組み合わせ回路と順序回路
- コンピュータのしくみ

## デジタル回路の実際

### ○ ロジックゲート

論理を実体化する回路素子

| A | B | NOT | AND | OR | NAND | NOR | XOR | XNOR |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | | 0 | 1 | 1 | 0 | 1 | 0 |
| 1 | 0 | | 0 | 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |



※NOTは一般に
インバータと呼ぶ

※JISでは新しい記号が制定されているが、これらが現役

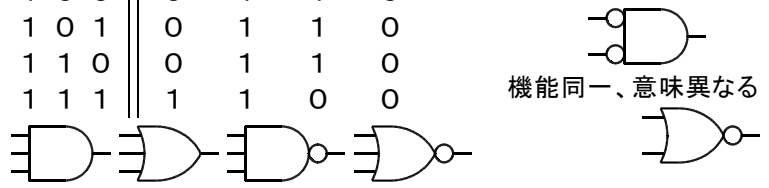


## デジタル回路の実際

### ○ ロジックゲート（多入力、入力否定）

| A | B | C | AND | OR | NAND | NOR |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 | 0 | 0 |

| A | B | $\overline{A}$ | $\overline{B}$ | $\overline{A}\cdot\overline{B}$ | $\overline{A}+\overline{B}$ |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 |



機能同一、意味異なる



## デジタル回路の実際

### ○ 組み合わせ回路 （と順序回路）

◇ 入力の0/1の組み合わせだけで
出力が決定される回路

vs 過去の入力の影響も受ける順序回路

＝真理値表で定義・説明できる回路

◇ 組み合わせ回路の設計

・1：とにかく真理値表を実現する

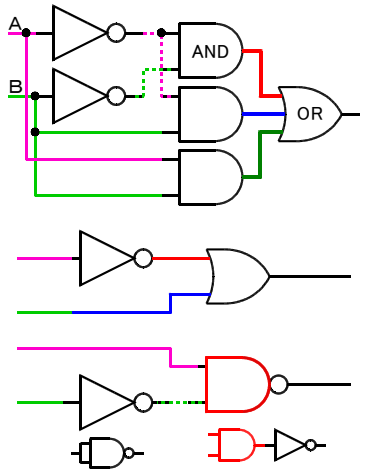
・2：最適化された回路（速くて安い）



## 組み合わせ回路

### ○ 複数の実体化

| A | B | 仕様 | 案1 | 案2 | 案3 |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 1,0,0 | 1,0 | 0,1 |
| 0 | 1 | 1 | 0,1,0 | 1,1 | 0,1 |
| 1 | 0 | 0 | 0,0,0 | 0,0 | 1,0 |
| 1 | 1 | 1 | 0,0,1 | 0,1 | 0,1 |




1： 個々の「1」の条件をANDで
作って、最後にORする

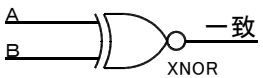
2： 「1 OR 1 =1」 なので「1」は
重なっても良い

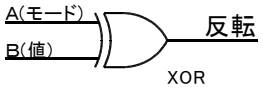
3： 最後にNOTする(NAND)
※2とド・モルガンの関係

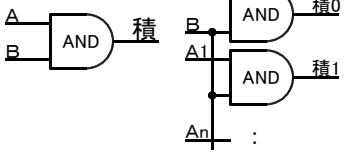
## 組み合わせ回路の実例

### ○ 一致検出　信号反転　乗算

| A | B | 一致 | 反転 | 乗算 |
|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 1 |




1：　2本の入力が一致すると1
2：　A＝0のときはBを出力
　　A＝1のときは$\overline{B}$を出力
3：　A×Bを出力する1ビット乗算
　　(A=1，B=1，(1×1=1)　のみ1)
　　並べれば1ビット×多ビット

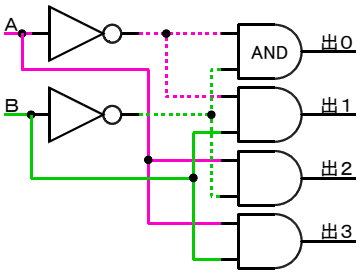


## 組み合わせ回路の実例

※参考：page36はこれをOR

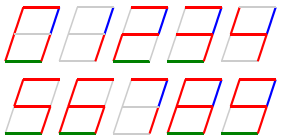
### ○ デコーダ

| A | B | 出0 | 出1 | 出2 | 出3 |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 |




入力に応じて、どれか1本だけが「1」になる。(一般に2進数解釈)

入力に応じて、特定のパターンを出力するデコーダもある。
例：2進数→数字（7セグデコーダ）



11111,00111　　10110,11011



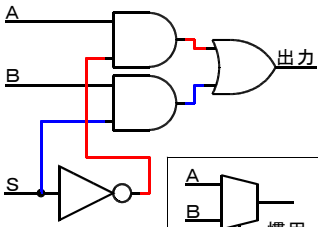
## 組み合わせ回路の実例

### ○ セレクタ

| S | A | B | 出力 |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

| S | A | B | 出力 |
|---|---|---|---|
| 0 | 0 | X | 0 |
| 0 | 1 | X | 1 |
| 1 | X | 0 | 0 |
| 1 | X | 1 | 1 |

| S | A | B | 出力 |
|---|---|---|---|
| 0 | X | X | A |
| 1 | X | X | B |




| S | B | AND | 通す |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | |
| 1 | 0 | 0 | B |
| 1 | 1 | 1 | |

S＝0ならAを出力
S＝1ならBを出力

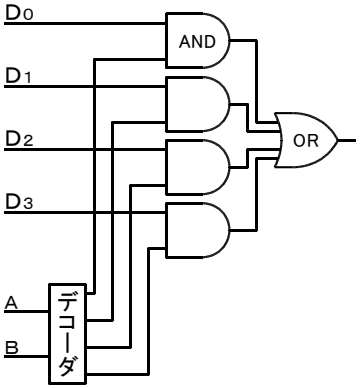


## 組み合わせ回路の実例

### ○ マルチプレクサ

| A | B | $D_0$ | $D_1$ | $D_2$ | $D_3$ | 出力 | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | X | X | X | 0 | $D_0$ |
| | | 1 | X | X | X | 1 | |
| 0 | 1 | X | 0 | X | X | 0 | $D_1$ |
| | | X | 1 | X | X | 1 | |
| 1 | 0 | X | X | 0 | X | 0 | $D_2$ |
| | | X | X | 1 | X | 1 | |
| 1 | 1 | X | X | X | 0 | 0 | $D_3$ |
| | | X | X | X | 1 | 1 | |

(A,B)で選択される入力4本のうち1本の状態を出力する。



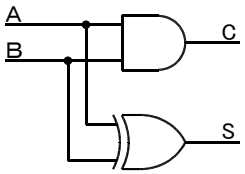

※セレクタは選択が1本、入力2本のマルチプレクサ

## 組み合わせ回路の実例

### ○ 半加算器

| 実現したい | A | B | C | S |
|---|---|---|---|---|
| 0＋0＝00 | 0 | 0 | 0 | 0 |
| 0＋1＝01 | 0 | 1 | 0 | 1 |
| 1＋0＝01 | 1 | 0 | 0 | 1 |
| 1＋1＝10 | 1 | 1 | 1 | 0 |

AND XOR




AとBを加算する。
S：和　　(sum)
C：繰り上がり　　(carry)

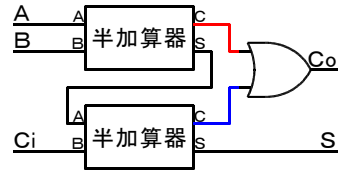
下の位からの繰り上がりに対応できないので「半」加算器。

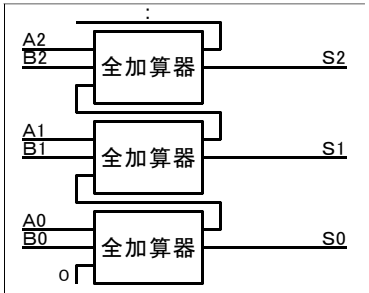


## 組み合わせ回路の実例

### ○ 全加算器

| 実現したい | Ci | A | B | Co | S |
|---|---|---|---|---|---|
| 0+0+0=00 | 0 | 0 | 0 | 0 | 0 |
| 0+0+1=01 | 0 | 0 | 1 | 0 | 1 |
| 0+1+0=01 | 0 | 1 | 0 | 0 | 1 |
| 0+1+1=10 | 0 | 1 | 1 | 1 | 0 |
| 1+0+0=01 | 1 | 0 | 0 | 0 | 1 |
| 1+0+1=10 | 1 | 0 | 1 | 1 | 0 |
| 1+1+0=10 | 1 | 1 | 0 | 1 | 0 |
| 1+1+1=11 | 1 | 1 | 1 | 1 | 1 |




下の桁からの繰り上がりCiに対応。
Ci+(A+B)、多段に繋ぐと多ビット化。



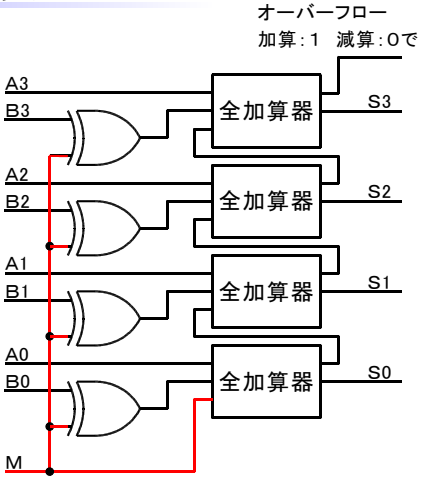
## 組み合わせ回路の実例

### ○ 加減算回路

M＝0：　加算（A+B）
M＝1：　減算（A−B）

→page 25,37

おさらい：
・×(−1)は「NOTして+1」
・反転はXORでできる
↓
・M=1のときXORでBをNOT
・Mを最下のCiに入れる
↓
たくさん積めば乗除算も。
※ただし遅い






## デジタル回路の実際

### ○ 組み合わせ回路のまとめ

◇ 実現したいルールを真理値表にする
　例）加算

◇ 組み合わせ回路の設計
　・1：とにかく真理値表を実現する
　・2：最適化された回路（速くて安い）

◇ 注意点
　・ゲートの処理には時間がかかる
　　入力→出力で ns($10^{-9}$)〜ps($10^{-12}$) 台

## デジタル回路の実際

○ 順序回路

◇ 過去の入力にも影響を受ける回路。
例）電卓　他ほとんど全てのデジタル機器

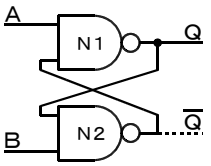
◇ 真理値表では表せない。
※擬似的に表す方法は回路によってはある
→ タイミングチャートを用いる
※信号の時間変化を表す図

◇ 組み合わせ回路とD－FF(後述)
によって構成されることが多い。

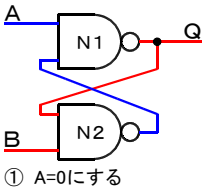


## 順序回路の実例

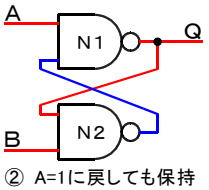
○ RSフリップフロップ (RS-FF)



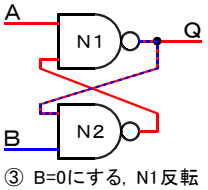

A,Bの入力は標準で1。
Aを0にすると出力Qは1。
Aを1に戻してもQ=1を維持。
Bを0にするとQ=0になる。
(N2出力が1になって、
N1出力が0になる→保持)



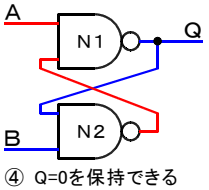
① A=0にする



② A=1に戻しても保持



③ B=0にする，N1反転

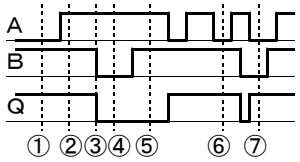


④ Q=0を保持できる

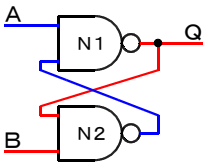


## 順序回路の実例

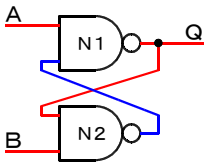
○ RSフリップフロップ (RS-FF)



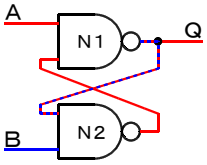

⑤ B=1にしてもQ=0を保持
⑥ すでにQ=1になっている
ときにA=0にしても変化無
⑦ A,Bともに0にすると、Qと
$\overline{Q}$はともに1に。A,Bのうち
後まで0のほうで決定。



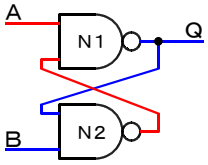
① A=0にする



② A=1に戻しても保持



③ B=0にする，N1反転

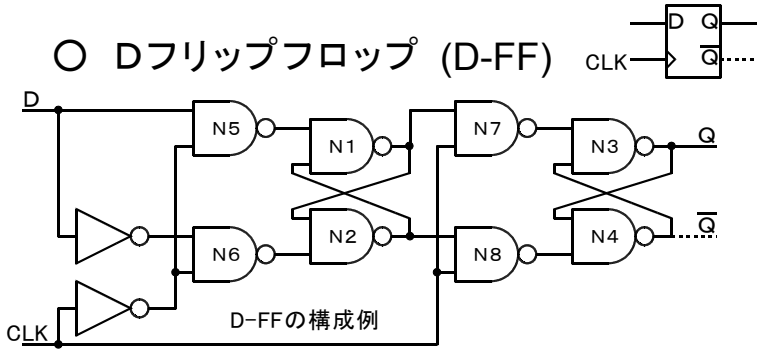


④ Q=0を保持できる

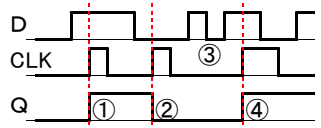


## 順序回路の実例

○ Dフリップフロップ (D-FF)

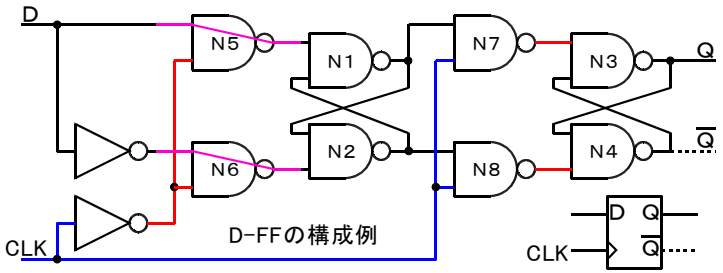



CLKを「0→1にした瞬間」(立ち上がり)
に、Dの値をQに反映、記憶する。
:①、②、④
それ以外はDは影響しない。

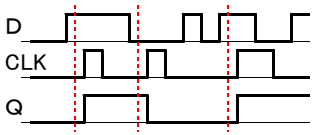


## 順序回路の実例

### ○ Dフリップフロップ (D-FF)



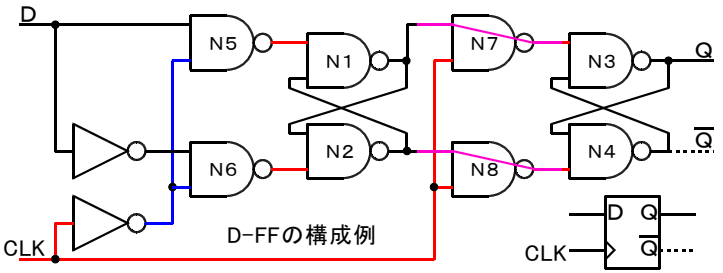

CLKが0の場合
NAND N5, N6はDを反映する。
よって、N1-N2のRSFFがDを記憶。
N7, N8は出力1でN3,N4は保持。



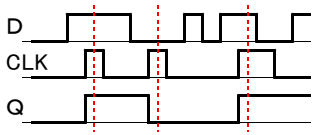



## 順序回路の実例

### ○ Dフリップフロップ (D-FF)



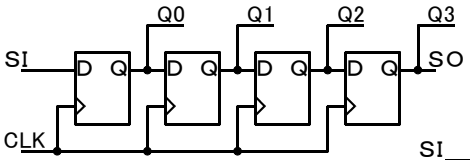

CLKが1の場合
N5,N6=1になって、N1-N2はDに影響
されず→CLK=1の直前を記憶。
N1-2がN7,8経由でN3-4に移される。





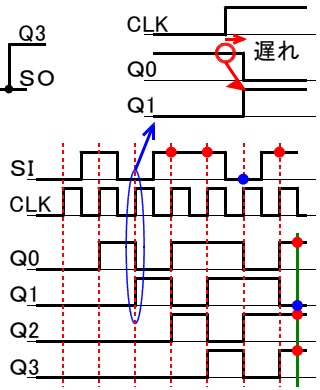

## 順序回路の実例：D-FFを用いた回路

### ○ シフトレジスタ




CLK↑のたびに、入力SIの状態を
取り込み、Q0～Q3に順番に送る。
順番に届くデータ(シリアル・・・・)を、
Q3～Q0に並べる(パラレル)。

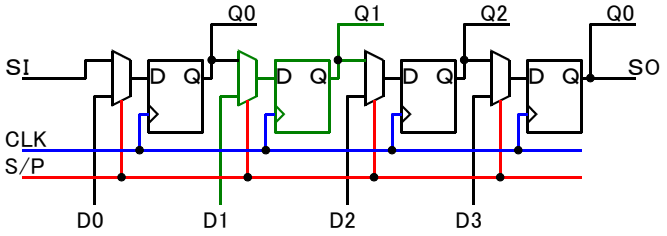
用途：通信






## 順序回路の実例：D-FFを用いた回路

### ○ シフトレジスタ(並直)

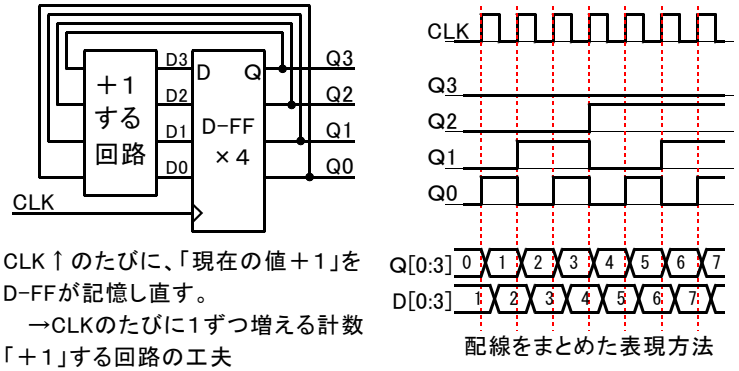



D-FFの入力にセレクタを追加、S/Pを0か1かで
「前段の出力」 か 「外部入力D0～D3」 か
を選択してCLK↑でD-FFに。
一度D0～D3をD-FFに入れてから、シフトさせると
並列データを直列にして出せる。

補足：
DとQをともに持つ
場合は少ない。
左右にシフトできる
ものもある。

## 順序回路の実例：D-FFを用いた回路

○ カウンタ（同期カウンタ）




CLK↑のたびに、「現在の値＋1」をD-FFが記憶し直す。
　→CLKのたびに1ずつ増える計数

「＋1」する回路の工夫
　→[カウントする/しない][上限値][up/down][初期設定]等可。

配線をまとめた表現方法



## デジタル回路の実際

○ 順序回路のまとめ

◇ 入力の履歴で出力が決まる。

◇ 実用的な回路の多くは
　［組み合わせ回路］ ＋ ［D-FF］
　で出来ている。（＝順序でも、組み合わせが基本）

◇ 注意点
　・一般に電源を入れた後の「初期状態」は未定。「初期化」「リセット」が必要。
　※ここまでの説明では「初期値0」を暗黙に。



## コンピュータのしくみ（超概要）

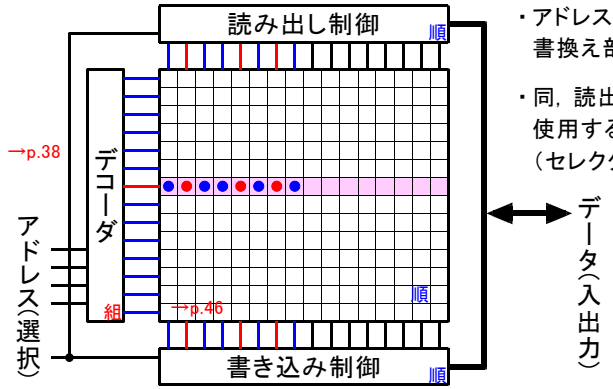
○ コンピュータは巨大な順序回路

◇ 記憶回路
◇ 演算回路
◇ 動作ステート
◇ 命令デコード
◇ 入出力回路
　：



## コンピュータのしくみ

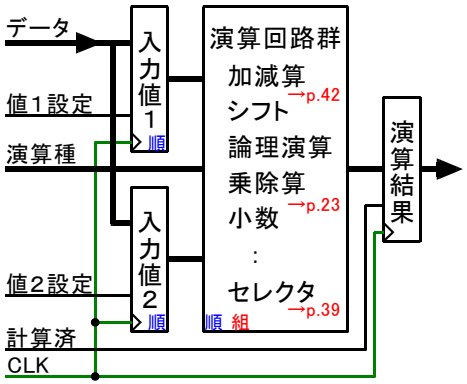
○ 記憶回路




・アドレスの一部で行を選択(デコーダ)
・アドレスの一部で書換え部位を選択
・同，読出した値から使用する部分を選択（セレクタ）

## コンピュータのしくみ

### ○ 演算回路




・記憶回路からデータを入力用レジスタに設定。
※DFFを並べたもの

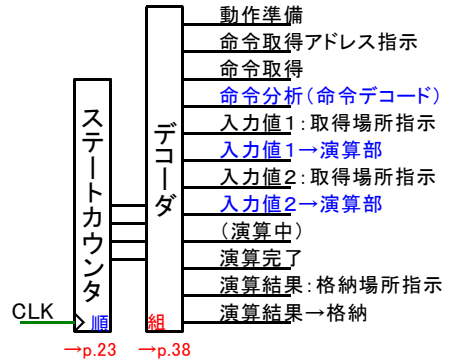
・コンピュータの命令の一部から演算種類を選択する。
※回路群＋セレクタなど

・演算が終わるころを見計らって結果を確保。



## コンピュータのしくみ

### ○ 動作ステート




・コンピュータは「手順」で動作している。
※プログラムという意味
※ではなく、回路として

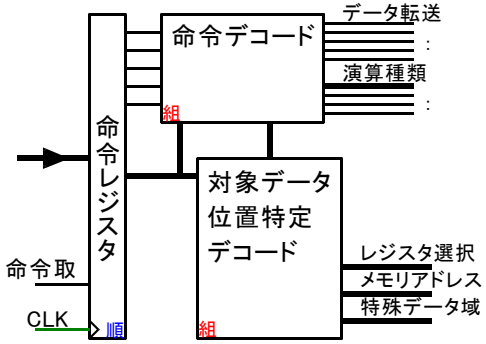
・カウンタ＋デコーダで作るよりは、一体で設計されていることが多。

・ここで作る信号は、各回路へのCLKではなく、各回路の動作開始信号。
※ないと回路は休止
CLKは全回路で共通。



## コンピュータのしくみ

### ○ 命令デコード




・コンピュータの命令は、単なるビット列(0/1の組)。

・それを「命令として」解釈する。
※そのルールに従って
※プログラムが作られる

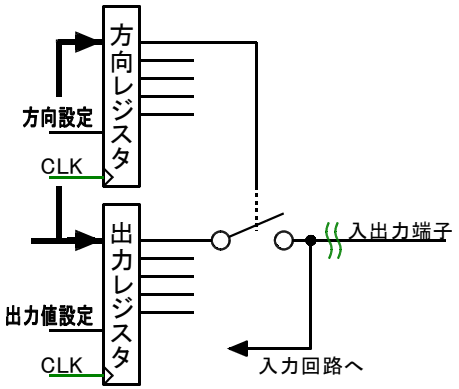
・ビット列の一部は、命令の種類を決定するのに使われる。
例）上位が000=移動
001=加算‥

・残りの部分で、演算等の対象を決める。
メモリ、CPU内レジスタ等



## コンピュータのしくみ

### ○ 信号出力回路




・出力回路は、内部で保持された信号をそのまま外に引き出す/加工する。

・組込マイコンでは、端子ごとに入出力方向を決定できる。

・出力端子に設定すると内部のスイッチがオンして出力値が外に出るように。
※スイッチには
※「スリーステート型」
※バッファが使われる
→p.63

## コンピュータのしくみ

○ コンピュータは巨大な順序回路

◇ 同一クロック信号による同期回路
＝いわゆる、パソコンの「クロック」

◇ 順序回路にはリセットが必要
＝ 各種LSIにリセット信号端子
& 電源オン時にリセット信号の回路
& リセットボタン（最近見なくなった）

◇ 仕様があれば気合いと根性？



## 本日のまとめ

○ デジタル回路は少しのルールと根性

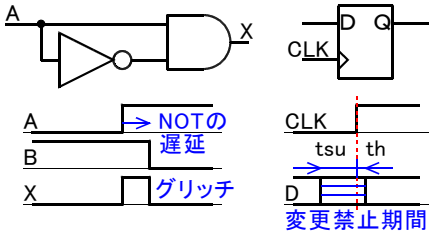
◇ 0 と 1 と AND と OR と NOT
・簡単な演算規則で、「ルール」に従った処理を行う回路を構成する。
・2進数はその「ルール」の一つ。

◇ 組み合わせ回路と順序回路
・組合わせ：そのときの入力だけで決まる。
・順序回路：過去の影響も受ける。
順序が大半でも組み合わせが基礎。



## 補遺：ロジック回路のプラスアルファ

○ 動作速度に起因する注意

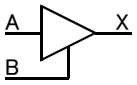






ロジックゲートには遅延がある。低速の回路では気にならないことが多いが、遅延が原因で予期せぬ「ヒゲ」(グリッチ)が発生することがある。(左上)
D-FF由来の回路には、入力信号変更禁止期間がある。tsu：セットアップ時間、th:ホールド時間。(右上) マイコン駆動時にも注意。

○ スリーステート



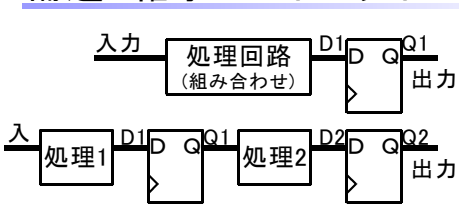

| B | A | X |
|---|---|---|
| 0 | 0 | Z |
| 0 | 1 | Z |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

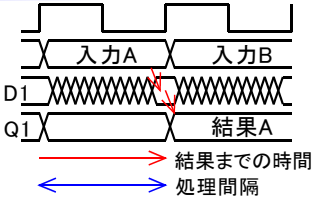
B=1のときX=A
B=0のときオフ(Z)

通常のゲートは常に0か1を出力するが、追加の制御線で「オフ」にできるものもある(P60のスイッチに相当)。
真理値表でオフは「Z」(Hi-Z等)で表記されることが多い。信号線の共有(バスなど)で多用されている。



## 補遺：雑学：パイプライン処理




回路の遅延によって複雑な組み合わせ回路の出力はバタつき(XXX区間)、後に落ち着く。落ち着いたところでDFFで結果を確定保持することが一般的(さもなければ、次の入力がくると前の結果が使えなくなる)。
ここで処理回路を敢えて分割すると、右下図のように処理結果が出てくる頻度が高くなる＝処理能力があがる。パイプライン処理といい、CPUはじめ高速デジタル処理回路でよく使われる手法。(このCLKが3GHzなどになる)