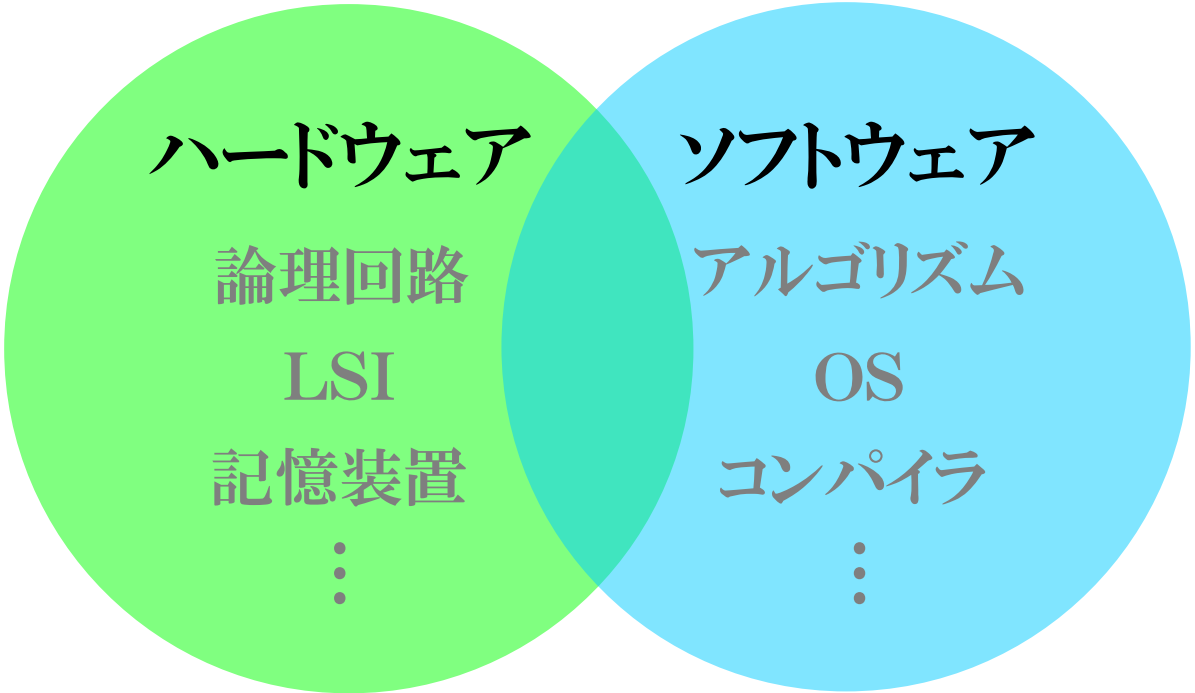
コンピュータ工学 I

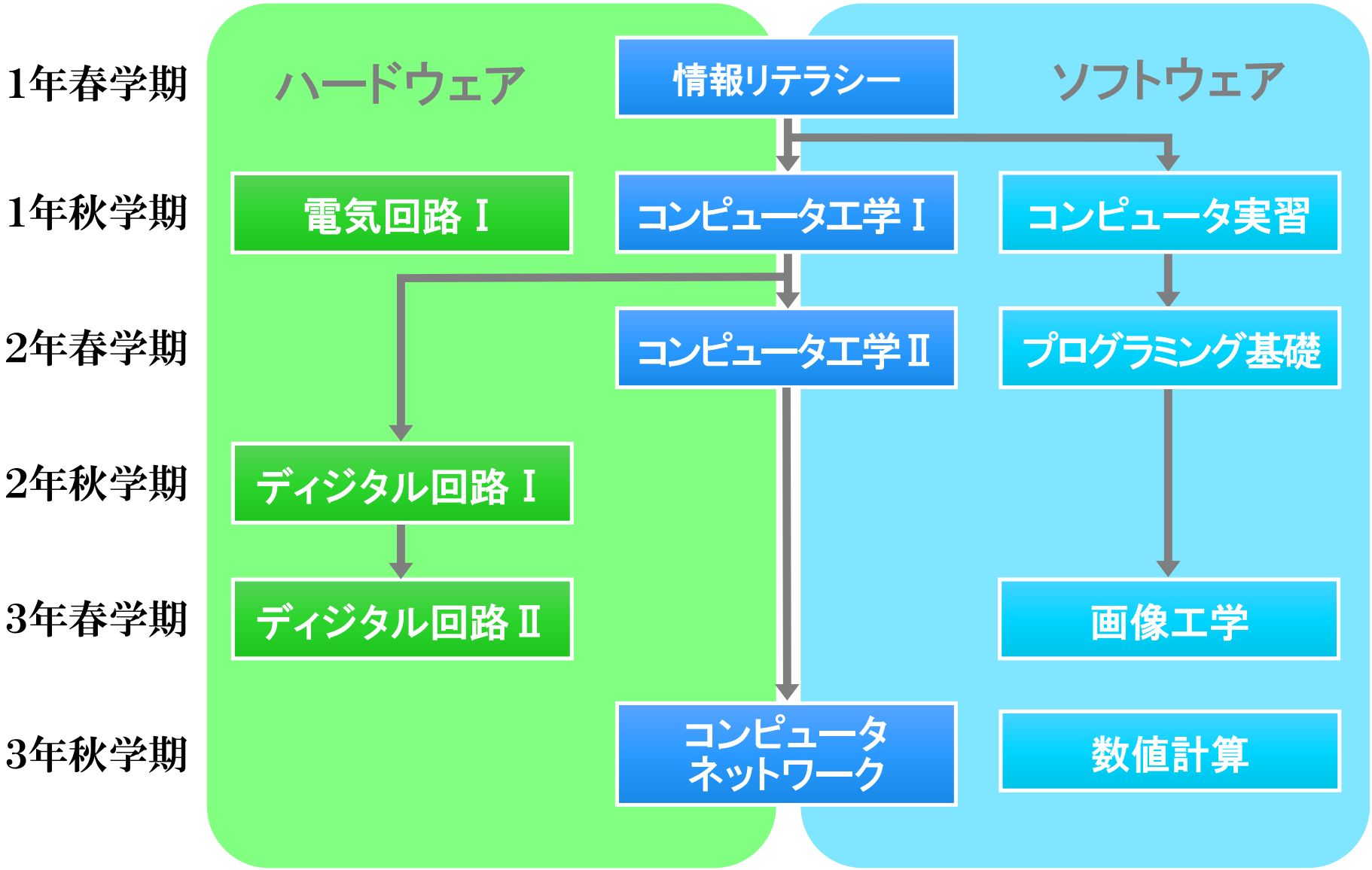
Rev. 2018.12.03

コンピュータ工学とは

コンピュータを実現するための技術について、ハードウェアとソフトウェアの両面から研究する学問。



カリキュラム体系


1年春学期
1年秋学期
2年春学期
2年秋学期
3年春学期
3年秋学期
ハードウェア
ソフトウェア
情報リテラシー
電気回路Ⅰ
コンピュータ工学Ⅰ
コンピュータ実習
コンピュータ工学Ⅱ
プログラミング基礎
ディジタル回路Ⅰ
ディジタル回路Ⅱ
画像工学
コンピュータ
ネットワーク
数値計算

コンピュータ工学Ⅰの内容

✤学習の目標

CPUの構造と動作の仕組みを理解する。

✤学習の内容

1. 数と文字の表現方法
2. 論理回路の設計
3. CPUの構成要素
4. アセンブリ言語とCPUの動作

レポート課題、試験について

✤ 確認テスト

毎回の講義後に、MOMO Campusで確認テストに解答すること。（1週間以内）

✤ レポート課題 全3回

✤ 試験

中間評価試験（第9回講義中）

最終評価試験（第16回）

数と文字の表現方法

✣ 内容

1. 2進数、16進数、基数変換
2. 負の数の表現方法
3. 演算（加算、減算、シフト演算）
4. 実数の表現方法
5. 文字の表現方法

データの記憶・伝達

電気の2状態　ON　OFF

2進数　1　0

10111000111001111110101000001101

2進数に変換

数値化

音声データ

画像データ

文章データ

数値データ

2018　2.718281828

−322　6.0221×10^{23}

数をかぞえる

2進数　　0 1

4進数　　0 1 2 3

8進数　　0 1 2 3 4 5 6 7

10進数　0 1 2 3 4 5 6 7 8 9

16進数　0 1 2 3 4 5 6 7 8 9 A B C D E F

数字として使う

2進数表記では桁数が多くなるので、便宜上、16進数で表記して桁数を少なくする。

進数対応表①

10進数	2進数	4進数	8進数	16進数
0	0	0	0	0
1				
2				
3				
4				
5				
6				
7				
8				

進数対応表①

10進数	2進数	4進数	8進数	16進数
0	0	0	0	0
1	1	1	1	1
2	10	2	2	2
3	11	3	3	3
4	100	10	4	4
5	101	11	5	5
6	110	12	6	6
7	111	13	7	7
8	1000	20	10	8

桁上がり

進数対応表②

10進数	2進数	4進数	8進数	16進数
8	1000	20	10	8
9	1001	21	11	9
10	1010	22	12	A
11	1011	23	13	B
12	1100	30	14	C
13	1101	31	15	D
14	1110	32	16	E
15	1111	33	17	F
16	10000	100	20	10

桁上がり

進数対応表③

10進数	2進数	4進数	8進数	16進数
16	10000	100	20	10
17	10001	101	21	11
18	10010	102	22	12
19	10011	103	23	13
20	10100	110	24	14
21	10101	111	25	15
22	10110	112	26	16
23	10111	113	27	17
24	11000	120	30	18

進数対応表④

10進数	2進数	4進数	8進数	16進数
24	11000	120	30	18
25	11001	121	31	19
26	11010	122	32	1A
27	11011	123	33	1B
28	11100	130	34	1C
29	11101	131	35	1D
30	11110	132	36	1E
31	11111	133	37	1F
32	100000	200	40	20

2、4、8、16進数の相互変換

2進数 2桁 ⟷ 4進数 1桁

2進数 3桁 ⟷ 8進数 1桁

2進数 4桁 ⟷ 16進数 1桁

1 1 1 0 1 → 131

1 1 1 0 1 → 35

1 1 1 0 1 → 1D

数の表現

✤ 数の表記

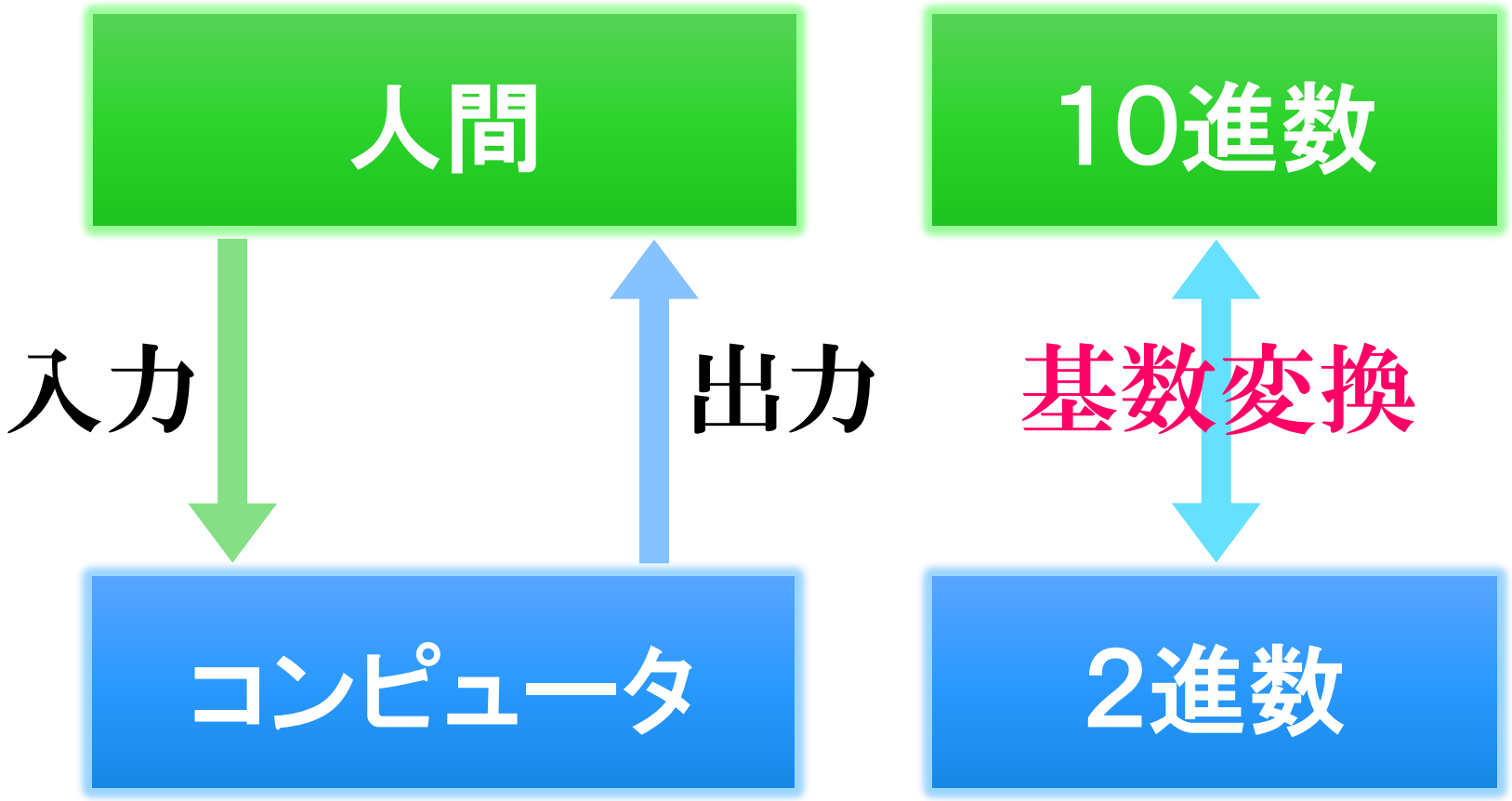
$N_{(p)}$：p進数の数値N

基数

✤ 基数変換

ある基数の数値を、別の基数の表記に変える。

基数変換の必要性



p進数整数部→10進数

10進数を$N_{(10)}$、

p進数整数部を $d_n d_{n-1} \cdots d_1 d_0\ _{(p)}$ と

する。$(n \geqq 0)$

$$N = d_n \cdot p^n + d_{n-1} \cdot p^{n-1} + \cdots + d_1 \cdot p^1 + d_0$$

p進数小数部→10進数

p進数小数部を $0.d_{-1}d_{-2}\cdots d_{m\,(p)}$ とする。$(m<0)$

$$N = d_{-1}\cdot p^{-1} + d_{-2}\cdot p^{-2} + \cdots + d_m \cdot p^m$$

p進数実数→10進数

重要

10進数を$N_{(10)}$、p進数を

$d_n d_{n-1} \cdots d_1 d_0 \bullet d_{-1} d_{-2} \cdots d_{m+1} d_m\ (p)$

とする。$(n \geqq 0 > m)$

$$N = d_n \cdot p^n + d_{n-1} \cdot p^{n-1} + \cdots + d_1 \cdot p^1 + d_0$$
$$+ d_{-1} \cdot p^{-1} + d_{-2} \cdot p^{-2} + \cdots + d_m \cdot p^m$$

10進数整数部→p進数

重要

p) $N_{(10)}$ 余り

p) q_0 … d_0

p) q_1 … d_1

⋮ ⋮

p) q_{n-1} … d_{n-1}

0 … $d_n = q_{n-1}$

pで割ったときの余りを求めることを繰り返す。

$$N_{(10)} = d_n d_{n-1} \cdots d_1 d_0\ (p)$$

原理

$$N_{(10)} \leftrightarrow d_n d_{n-1} \cdots d_2 d_1 d_0\ (p)$$

$$N$$
$$= d_n \cdot p^n + d_{n-1} \cdot p^{n-1} + \cdots + d_2 \cdot p^2 + d_1 \cdot p + d_0$$
$$= p \cdot (d_n \cdot p^{n-1} + d_{n-1} \cdot p^{n-2} + \cdots + d_2 \cdot p + d_1) + d_0$$

$N \div p$の商

$N \div p$の余り

N を p で繰り返し割っていけば、
その余りから $d_0, d_1, \cdots, d_n$ が順番に求まる。

10進数小数部→p進数

重要

$$N \times p = a_0 + b_0$$

$$b_0 \times p = a_1 + b_1$$

$$b_1 \times p = a_2 + b_2$$

$$\vdots$$

$b_n = 0$ になるまで続ける。

$0 < N_{(10)} < 1$

a_i 整数部

b_i 小数部

小数部にpを掛けることを繰り返す。

$$N_{(10)} = 0.\, a_0 a_1 \cdots a_n\ (p)$$

基数変換まとめ

重要

✣ p進数 → 10進数

p進数のk桁目の値d_kにp^kを掛けて総和を取る。1の位が0桁目になる。

✣ 10進数 → p進数

整数部をpで繰り返し割っていき、最後の余りを左端にして順に並べる。

小数部にpを繰り返し掛けていき、最初の整数値を左端にして順に並べる。

2進数

重要

MSB（最上位bit）

LSB（最下位bit）

1 0 1 0 1 (2)

bit　2進数の1桁

8 bit ＝ 1 byte

$n$bitの2進数は、2^n個の数を表現できる。

2進数の加算

✤基本演算

0	1	1	1
+ 0	+ 0	+ 1	+ 1
0	1	10	11

bit数の制限

コンピュータの記憶容量は有限

数値1個のbit数（桁数）に上限がある

- bit数が分かるように、MSBまでを0で埋めて表記する。
- MSBを越えた（オーバーフローした）bitの情報は捨てる。

シフト演算

重要

- 2進数の全bitを左または右に移動する。
- MSBまたはLSBを越えたbitの情報は捨てる。

$n$bit 左シフト → 2^n倍になる。

$n$bit 右シフト → $\frac{1}{2^n}$倍になる。

※MSBやLSBを越えない範囲で

2進数の乗算

シフト演算と加算の組み合わせで実現できる。

```
    111
  × 101
  -----
    111     111を0bit左シフト
+ 11100     111を2bit左シフト
  -----
 100011     答え
```

負の整数

コンピュータは０と１しか使えない。

↓

＋とーの符号が存在しない

✤負の数の表現方法

❶ 符号絶対値法

❷ 1の補数

❸ 2の補数

符号絶対値法

MSBを符号として用いる。

0 0 0 1 1 1

重要

符号bit

0 → ＋　正の数、または、0

1 → －　負の数

1の補数

✤ 1の補数とは

2進数 x の1の補数を $\overline{x}$ とするとき、

$$x + \overline{x} = 111\cdots1_{(2)}$$

となる。

✤ 求め方

重要

x の0を1に、1を0に置き換える。
（bit反転）

2の補数

✤ 2の補数とは

2進数 x の**2の補数**を x' とするとき、

$$x + x' = 100\cdots0_{(2)}$$

となる。

✤ 求め方

重要

x の**1の補数**に1を加算する。

負の数は、2の補数を用いて表す。

コンピュータでの整数の表現方法

$n$bitの2進数は、2^n個の数を表現できる。

✣ 符号なし2進数

MSBを符号bitとしない。

$$0 \leqq x \leqq 2^n - 1$$

✣ 符号つき2進数

MSBを符号bitとする。

$$-2^{n-1} \leqq x \leqq 2^{n-1} - 1$$

コンピュータにおける減算の方法

減算は、負の数との加算に置き換えて計算する。

$$5-3=$$

$$5+(-3)=$$

5＋(－3)の計算

$5_{(10)} = 0101_{(2)}$

$3_{(10)} = 0011_{(2)}$

$0011_{(2)}$の2の補数は

$1101_{(2)}$

したがって、

$-3_{(10)} = 1101_{(2)}$

$$\begin{array}{r} 0101 \\ +\ 1101 \\ \hline 10010 \end{array}$$

答えは $0010_{(2)}$

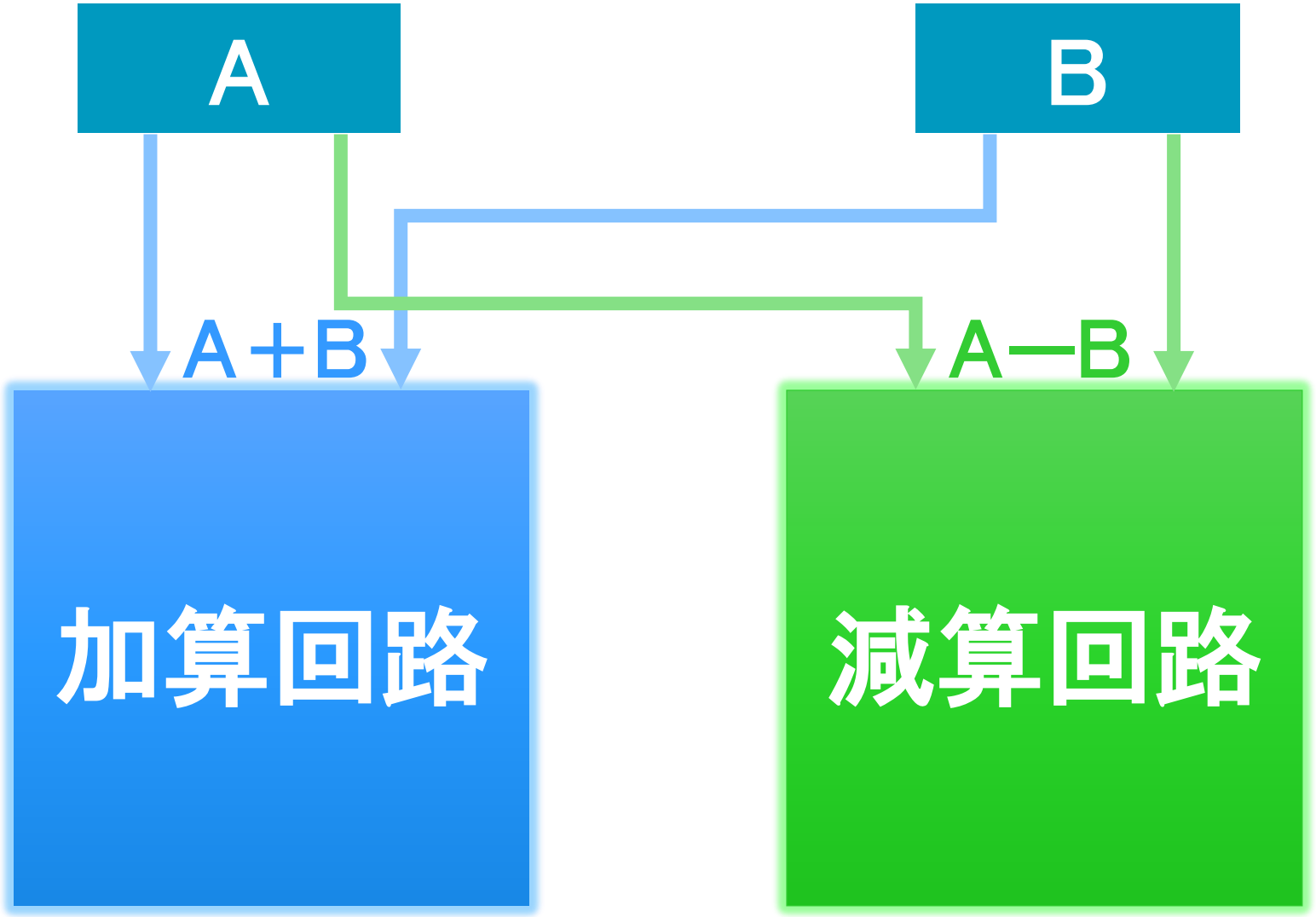
$= 2_{(10)}$

コンピュータ設計の方針

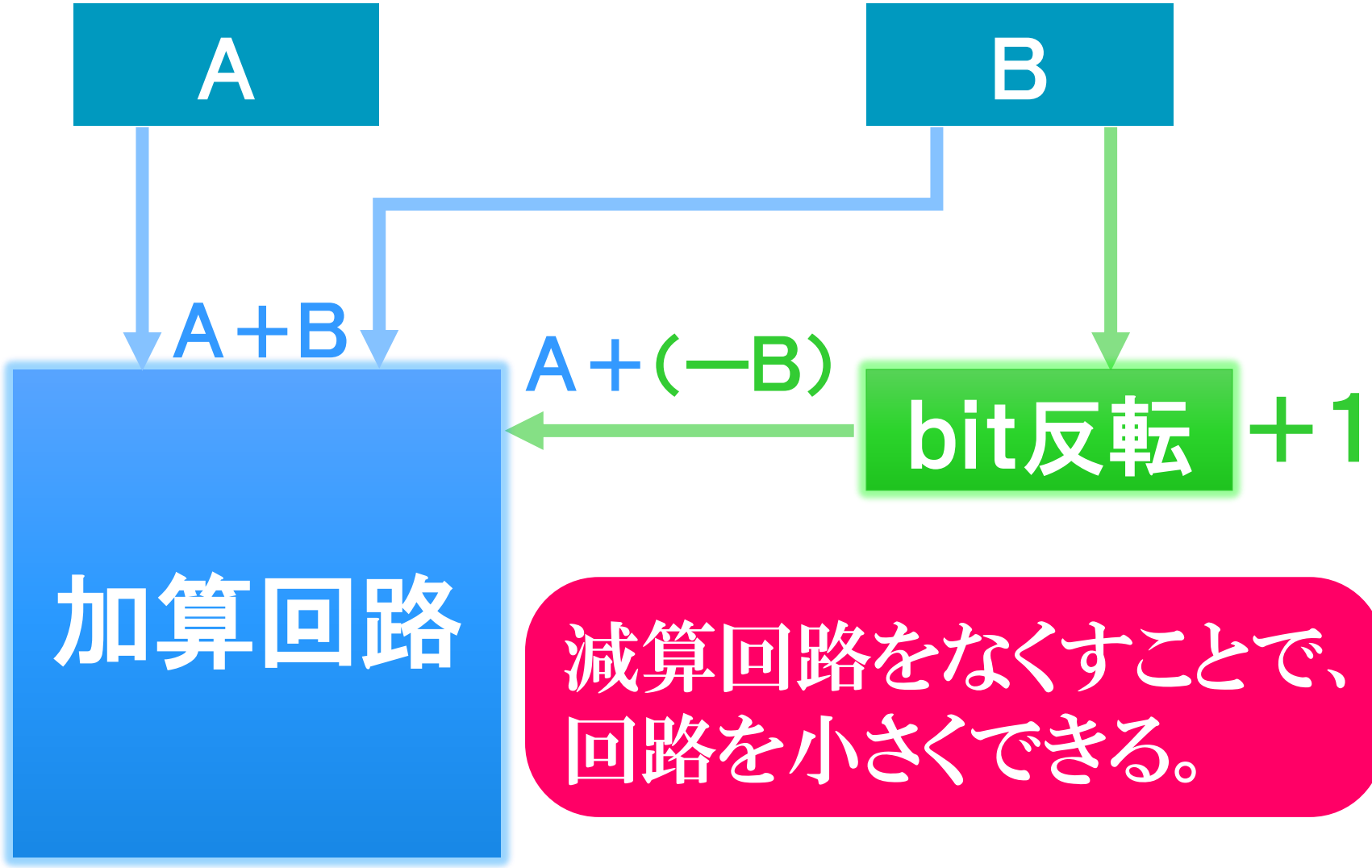
- 良いコンピュータとは？
  - 高速
  - 小型
  - 低コスト

コンピュータの回路の無駄を省き、可能な限り小さく構成する。

加減算回路の構成①



加減算回路の構成②



コンピュータでの実数の表現方法

✤ 固定小数点数

小数点の位置を特定のbit間に固定する。

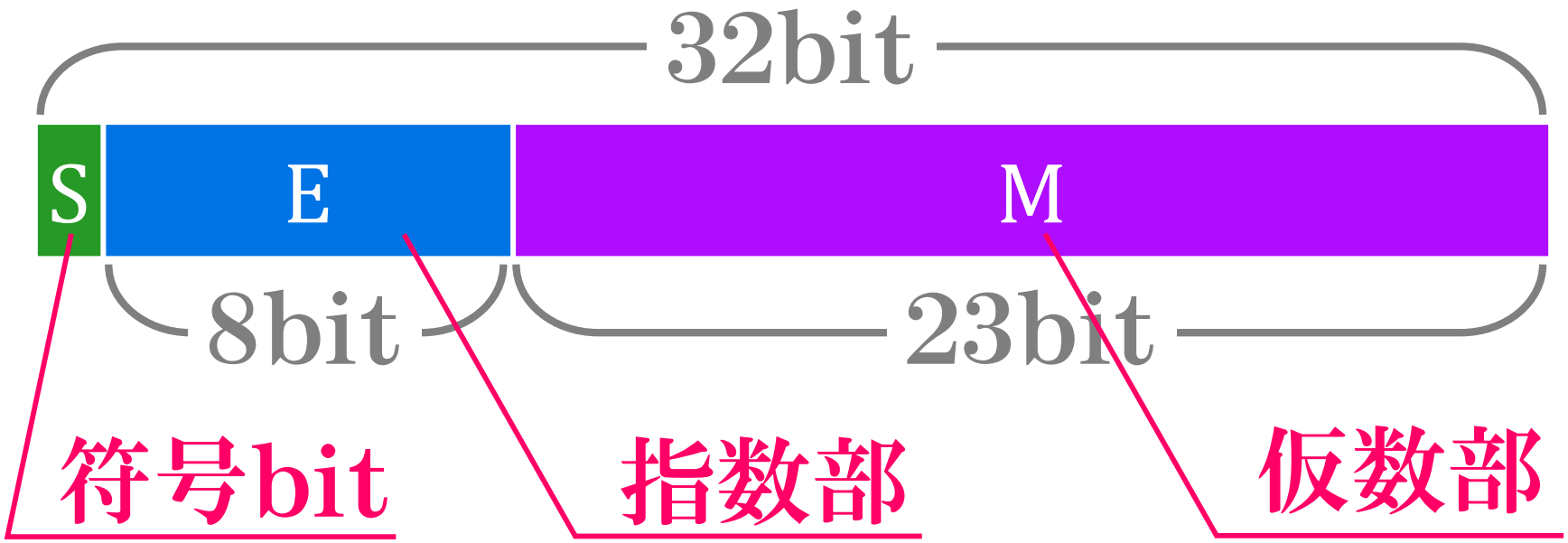
整数は、小数部のない固定小数点数

bit数より桁数の多い整数や小数が表現できない。

✤ 浮動小数点数

仮数部と指数部に分けて数値を表現する。

単精度浮動小数点数



$$N = (-1)^{S} \times 1.M_{(2)} \times 2^{E-127}$$

C言語の変数型（参考）

整数型（固定小数点数）

型名	bit数	表現できる数値の範囲
char	8	−128～127
short	16	−32,768～32,767
long (int)	32	−2,147,483,648～2,147,483,647

実数型（浮動小数点数）

型名	bit数	仮数のbit数	指数の範囲
float	32	23	2の−126～127乗
double	64	52	2の−1022～1023乗

整数(固定小数点数)同士の計算

$$
\begin{array}{r}
0011101 \\
+\ 0000110 \\
\hline
0100011
\end{array}
$$

bitの並びを変えずに計算ができる。

計算処理が単純 ➡ 計算が速い

実数(浮動小数点数)同士の計算

$$1.1101\times2^{5}$$
$$+\ 1.1000\times2^{3}$$

$$1.1101\times2^{5}$$
$$+\ 0.0110\times2^{5}$$
$$10.0011\times2^{5}$$

$$1.0001\times2^{6}$$

指数の値を揃えてから計算する。

計算処理が複雑 ➡ 計算が遅い

数値表現の長所と短所

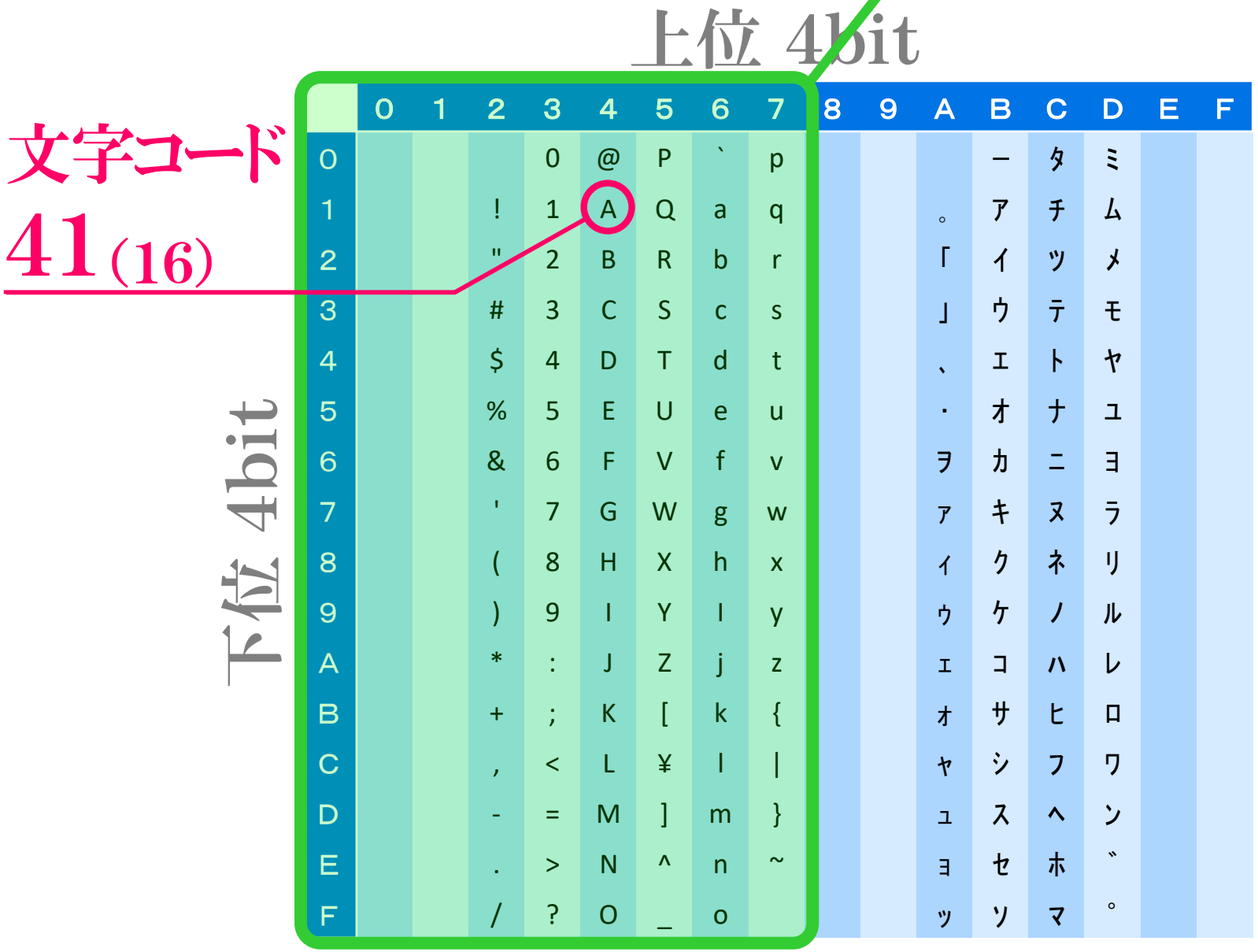
固定小数点数	浮動小数点数
表現できる数値の範囲が狭い	表現できる数値の範囲が広い
計算が速い	計算が遅い

誤差問題

- **丸め誤差**
  小数点以下の下位の桁を削除することにより誤差が生じる。
- **情報落ち**
  絶対値の大きい数と小さい数の加減算をするとき、小さい数が計算結果に反映されない。
- **桁落ち**
  ほぼ等しい数の減算をするとき、有効桁数が大幅に失われる。

8bit JISコード

ASCIIコード



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0				0	@	P	`	p				ー	タ	ミ		
1			!	1	A	Q	a	q			。	ア	チ	ム		
2			"	2	B	R	b	r			「	イ	ツ	メ		
3			#	3	C	S	c	s			」	ウ	テ	モ		
4			$	4	D	T	d	t			、	エ	ト	ヤ		
5			%	5	E	U	e	u			・	オ	ナ	ユ		
6			&	6	F	V	f	v			ヲ	カ	ニ	ヨ		
7			'	7	G	W	g	w			ァ	キ	ヌ	ラ		
8			(	8	H	X	h	x			ィ	ク	ネ	リ		
9			)	9	I	Y	I	y			ゥ	ケ	ノ	ル		
A			*	:	J	Z	j	z			ェ	コ	ハ	レ		
B			+	;	K	[	k	{			ォ	サ	ヒ	ロ		
C			,	<	L	¥	l	|			ャ	シ	フ	ワ		
D			-	=	M	]	m	}			ュ	ス	ヘ	ン		
E			.	>	N	^	n	~			ョ	セ	ホ	゛		
F			/	?	O	_	o				ッ	ソ	マ	゜		

16bit JISコード

16bit　　8bit

東京City

456C　357E　43　69　74　6F (16)

E1 5~ City

コードの種類を誤ると、文字化けが起こる。

16bit JISコード

16bit　　8bit

東京city

456C　357E　43　69　74　6F (16)

1B2442
16bitコード開始

1B284A
8bitコード開始

制御コードを
埋め込む

16bit シフトJISコード

16bit　　8bit

東京city

938C　8B9E

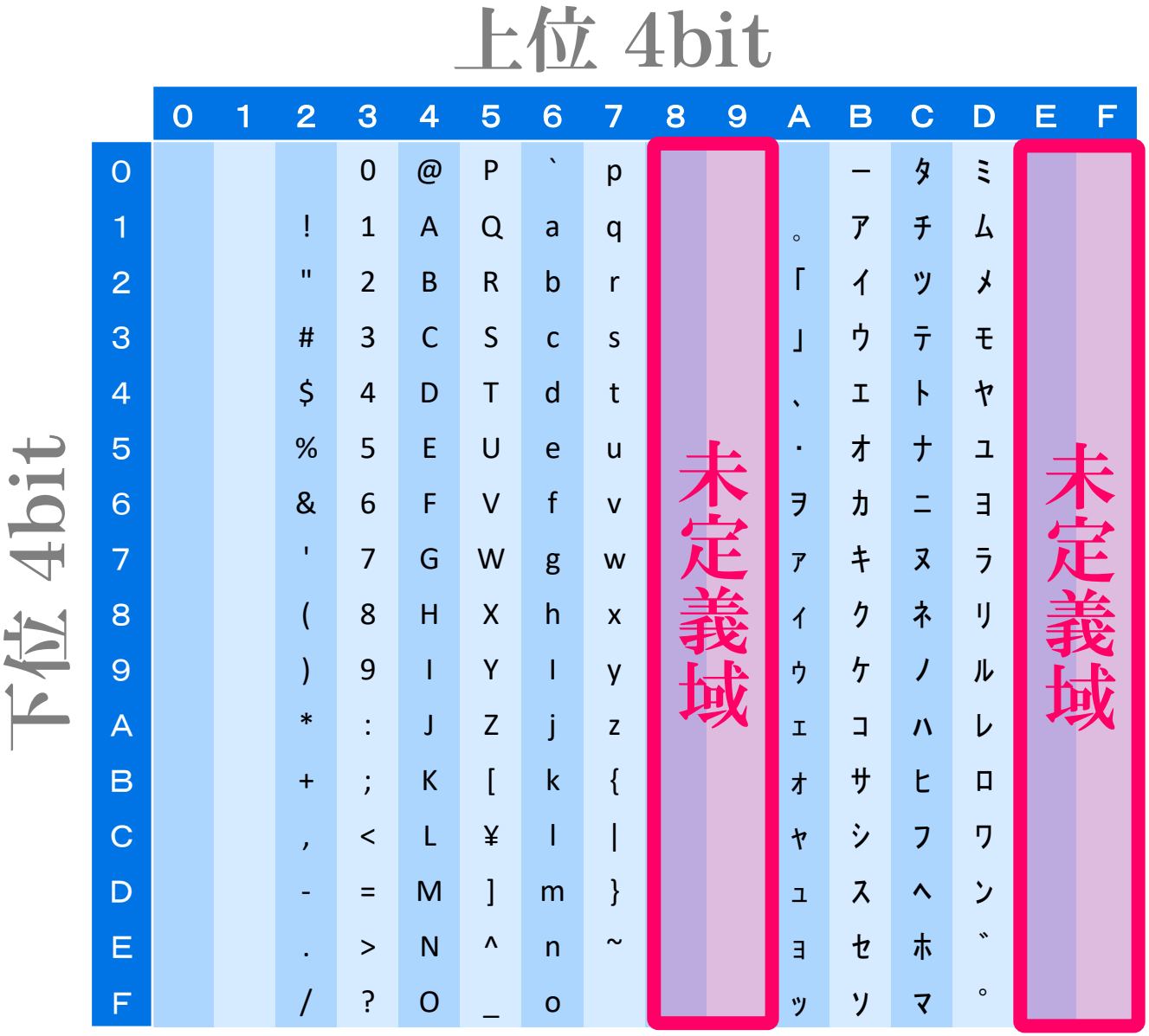
未定義　未定義

43 69 74 6F (16)

City

上位8bitに未定義域コードを使う。

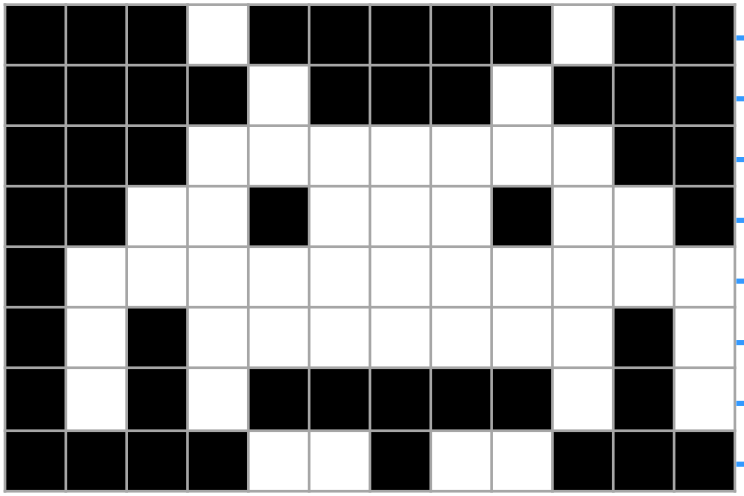
未定義域コードとは


上位 4bit
下位 4bit
0 1 2 3 4 5 6 7 8 9 A B C D E F
0 0 @ P ` p ー タ ミ
1 ! 1 A Q a q 。 ア チ ム
2 " 2 B R b r 「 イ ツ メ
3 # 3 C S c s 」 ウ テ モ
4 $ 4 D T d t 、 エ ト ヤ
5 % 5 E U e u ・ オ ナ ユ
6 & 6 F V f v ヲ カ ニ ヨ
7 ' 7 G W g w ァ キ ヌ ラ
8 (8 H X h x ィ ク ネ リ
9) 9 I Y I y ゥ ケ ノ ル
A * : J Z j z ェ コ ハ レ
B + ; K [k { ォ サ ヒ ロ
C , < L ¥ l | ャ シ フ ワ
D - = M] m } ュ ス ヘ ン
E . > N ^ n ~ ョ セ ホ ゛
F / ? O _ o ッ ソ マ ゜
未定義域
未定義域

ユニコード(UNICODE)

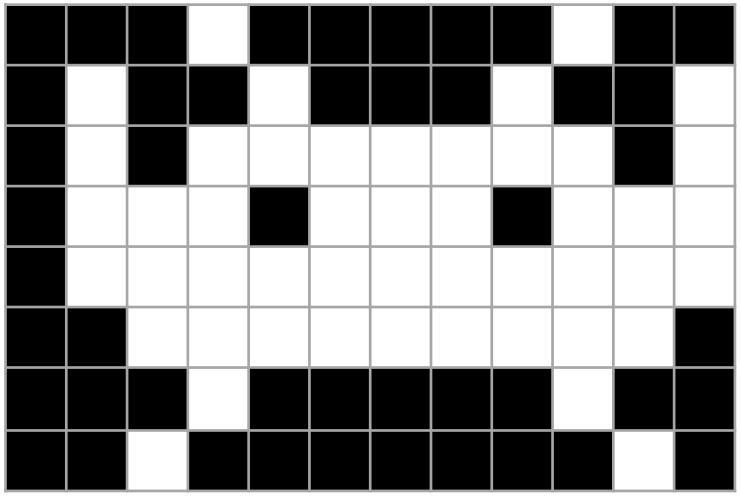
多国語に対応するため、世界中の主要な文字や記号をまとめた文字コード

ビットマップ（白黒画像）



0001 0000 0100 → 104
0000 1000 1000 → 088
0001 1111 1100 → 1FC
0011 0111 0110 → 376
0111 1111 1111 → 7FF
0101 1111 1101 → 5FD
0101 0000 0101 → 505
0000 1101 1000 → 0D8

黒を0、
白を1にする。

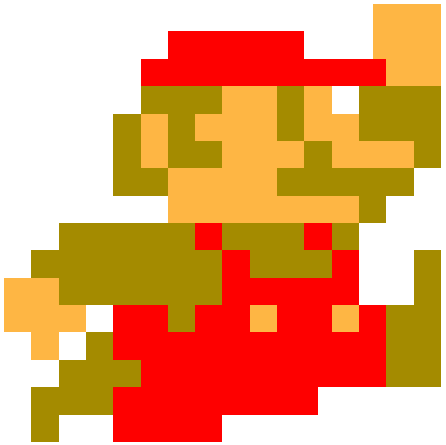


数値化された画像
1040881FC3767FF5FD5050D8

スペースインベーダー ©TAITO Corp. より

カラービットマップ（カラー画像）

1ドットに複数のbitを割り当て、多値を表現する。



スーパーマリオブラザーズ
©Nintendo より

ファミリーコンピュータ（任天堂,1983年）では、1ドットに 2bitを割り当てている。

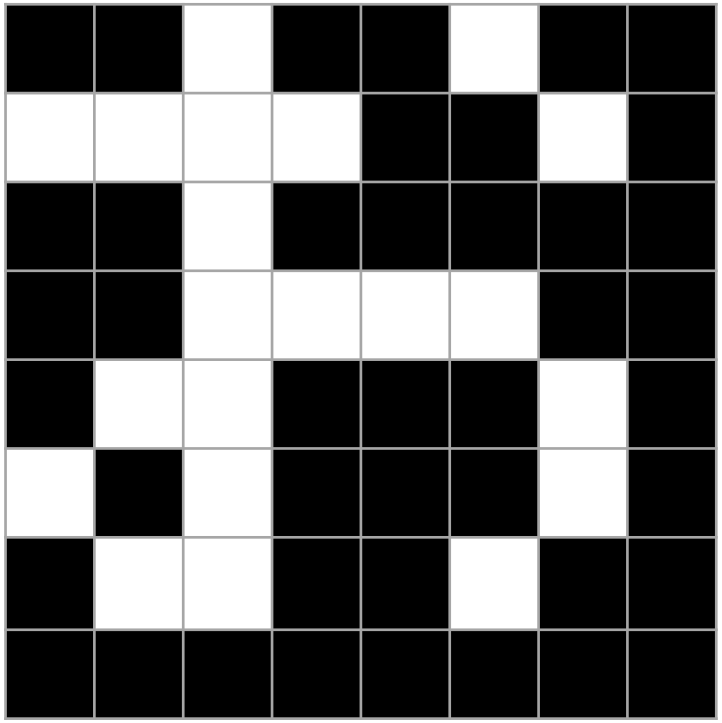
そのため、キャラクターは最大4色（そのうち1色は背景の透明色）で表現されている。（カラーパレット方式）

現在の一般的なコンピュータは、1ドットに24bitを割り当て、最大16,777,216色を表現できる。

ビットマップフォント

点の集合（ビットマップ）でつくられたフォント

文字「お」



デザイン参考：
8x8日本語フォント「美咲フォント」

0010 0100	24
1111 0010	F2
0010 0000	20
0011 1100	3C
0110 0010	62
1010 0010	A2
0110 0100	64
0000 0000	00

24F2203C62A26400

課題 1

- 教科書29ページ 演習問題
- これまでの講義についての感想

  どのような内容（感想・要望）でも良い。

提出日時：12月5日（水） 講義開始前

レポートの作成について

- 岡山理大学専用のレポート用紙に書く。
- ホッチキスまたは糊で綴じる。
- 学生番号、氏名、講義名、提出日を書く。
- 途中の計算過程を書く。
- 解答した後、教科書の正答を見て、赤ペンで○×をつける。
- 間違えた問題、解けなかった問題は赤ペンで計算過程と正答を書く。

論理回路

✣ 内容

1. 論理演算（ブール代数）
2. ブール代数の公理
3. 論理式の簡単化
4. 組み合わせ論理回路
5. 順序回路

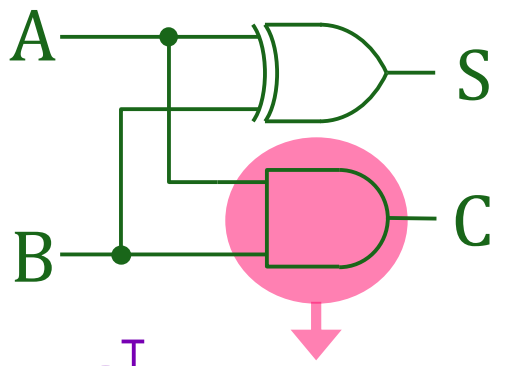
演算回路の設計

❶ 回路化したい計算式 $d = a + b$

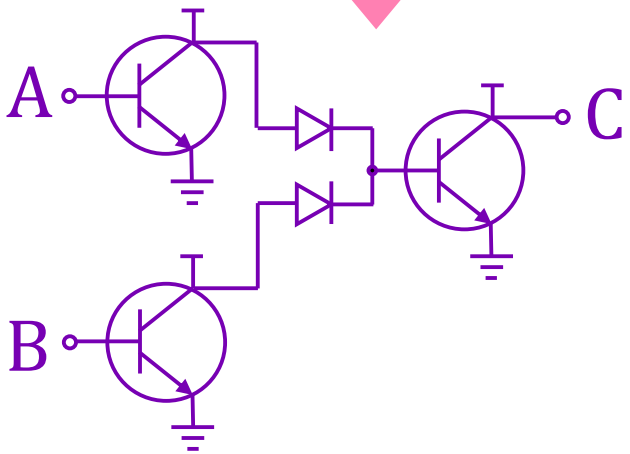
❷ 論理式

$$S = A \oplus B$$
$$C = A \cdot B$$

❸ 論理回路



❹ ディジタル回路



論理演算(ブール代数)

重要

真(true)と偽(false)の2状態を扱う演算

論理値

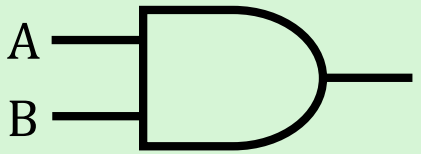
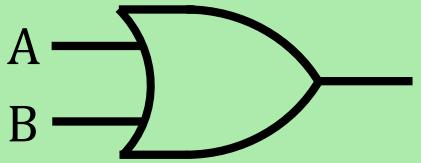
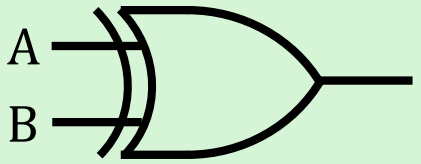
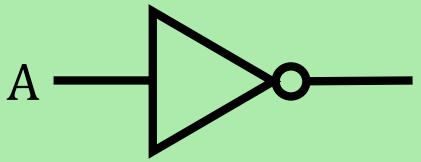
真 … 1

偽 … 0

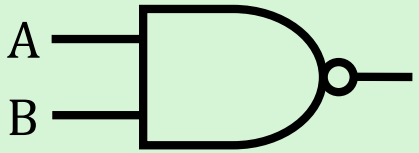
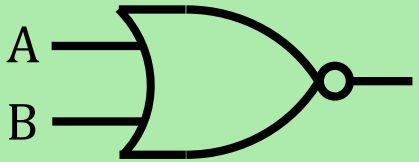
2進数(0, 1)の演算の実現に適している。

基本論理演算

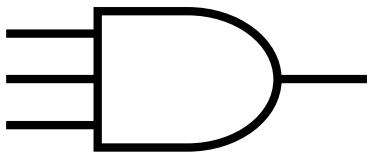
重要

	演算記号	回路記号
論理積 AND	$A \cdot B$	
論理和 OR	$A + B$	
排他的論理和 Exclusive OR （XOR）	$A \oplus B$	
否定 NOT	$\overline{A}$	

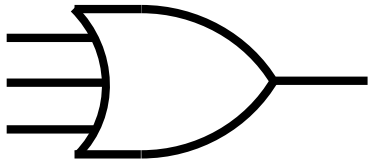
基本論理演算

	式の表記	回路記号
否定論理積 NAND	$\overline{A \cdot B}$	
否定論理和 NOR	$\overline{A + B}$	

3入力の場合



$A \cdot B \cdot C$



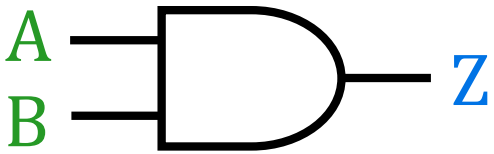
$A + B + C$

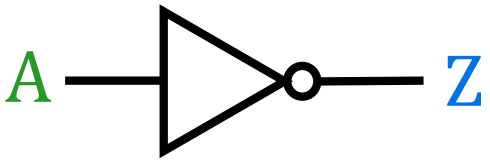
基本論理演算の真理値表

重要

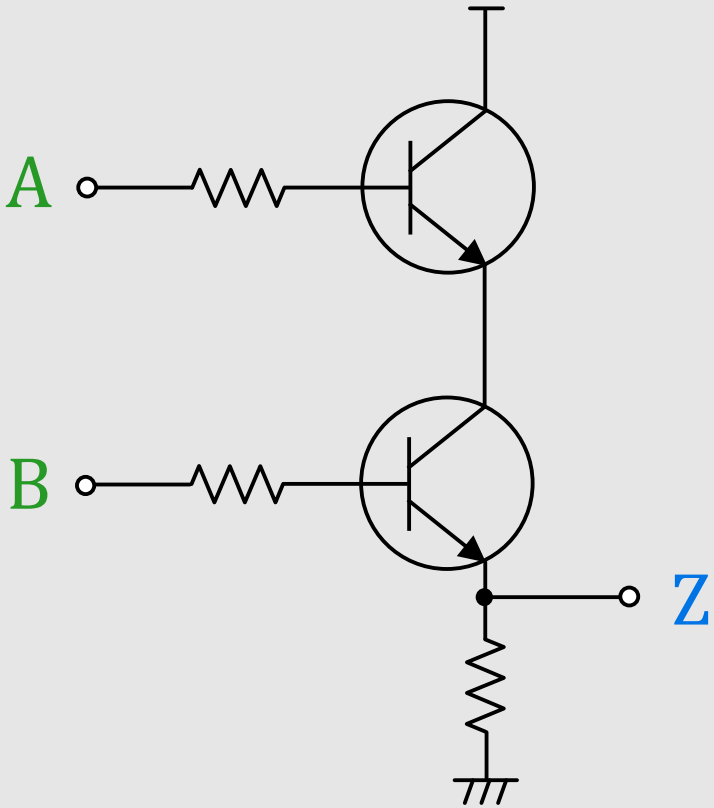
入力値		出力値			
A	B	$A \cdot B$	$A + B$	$A \oplus B$	$\overline{A}$
0	0	0	0	0	1
0	1	0	1	1	
1	0	0	1	1	0
1	1	1	1	0	

トランジスタによる論理回路の構成

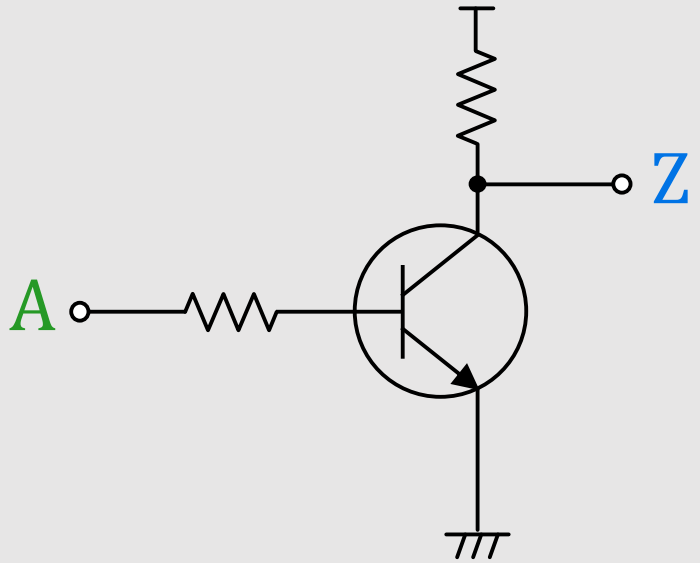




AND回路

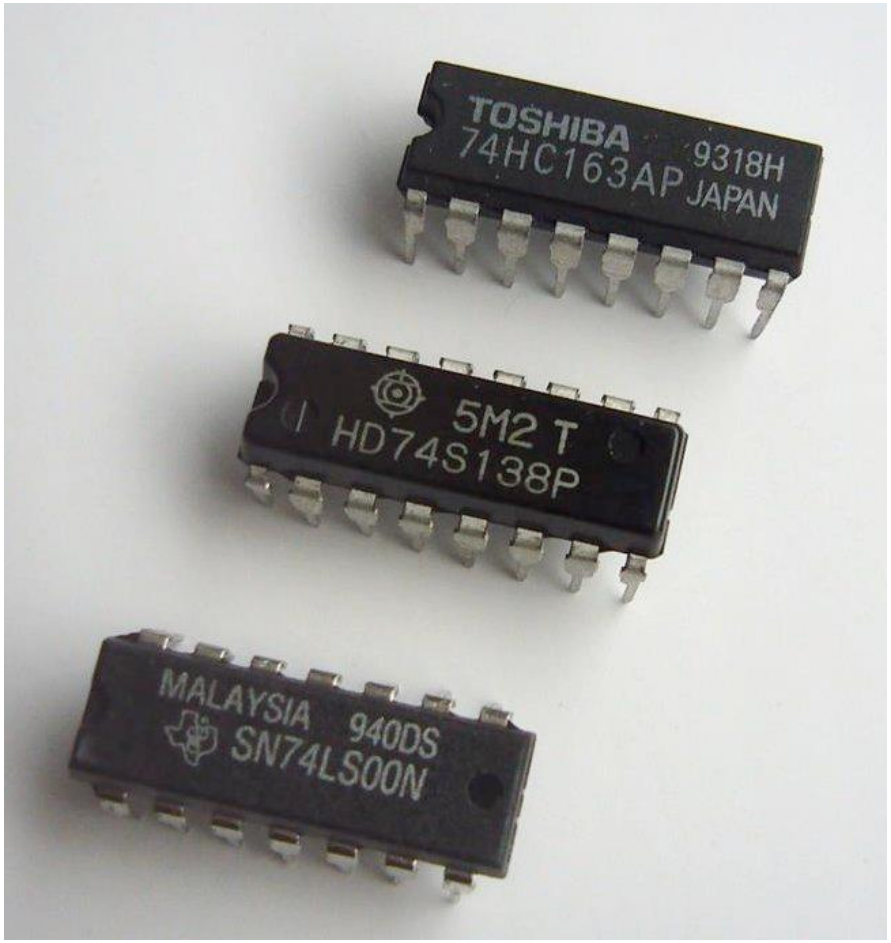


NOT回路

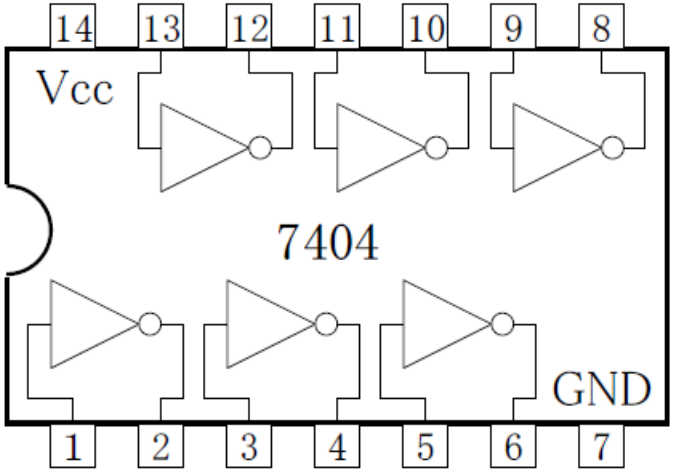


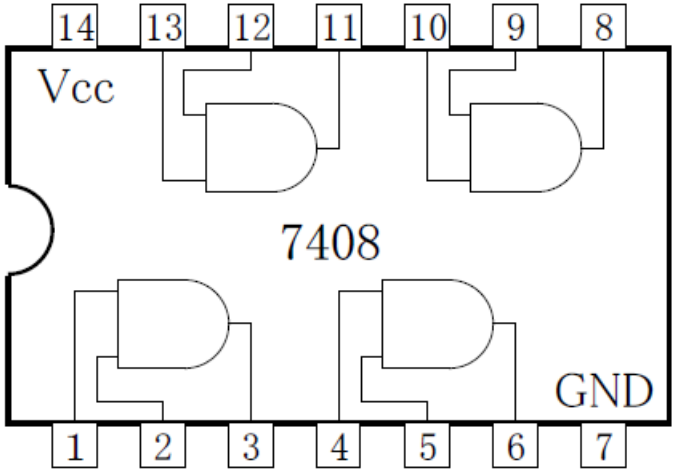
※ 回路構成の一例

ロジックIC（集積回路）



Wikipedia「汎用ロジックIC」より





ビット演算

```
  1001
AND 1010
----
  1000
```

```
  1001
OR  1010
----
  1011
```

```
  1001
XOR 1010
----
  0011
```

```
NOT 1001
----
 0110
```

C言語の論理演算子(参考)

ビット演算子

演算名	記号	例
AND	&	a & b
OR	|	a | b
XOR	^	a ^ b
NOT	~	~a

1bit ごとに論理演算を行う。

結合演算子

演算名	記号	例
AND	&&	a>0 && b>0
OR	||	a>0 || b>0
XOR	なし	
NOT	!	!(a>0)

非0を真、0を偽とみなして、論理演算を行う。

ブール代数の公理

重要

- べき等則
- 交換則
- 結合則
- 吸収則
- 分配則
- 二重否定
- ド・モルガン則
- 単位元
- 零元
- 補元

公式は別紙資料を参照せよ。

演算回路の作成手順

❶ 真理値表の作成

❷ 論理式の組み立て

❸ 論理式の簡単化

❹ 論理回路への置き換え

半加算器

重要

真理値表

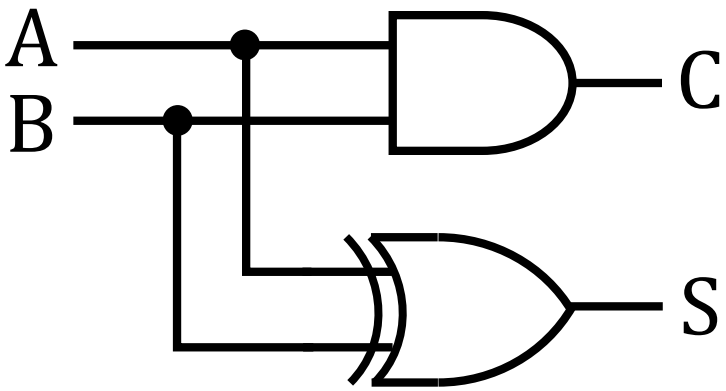
A	B	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

論理式

$$C = A \cdot B$$

$$S = A \oplus B$$

論理回路



全加算器

重要

真理値表

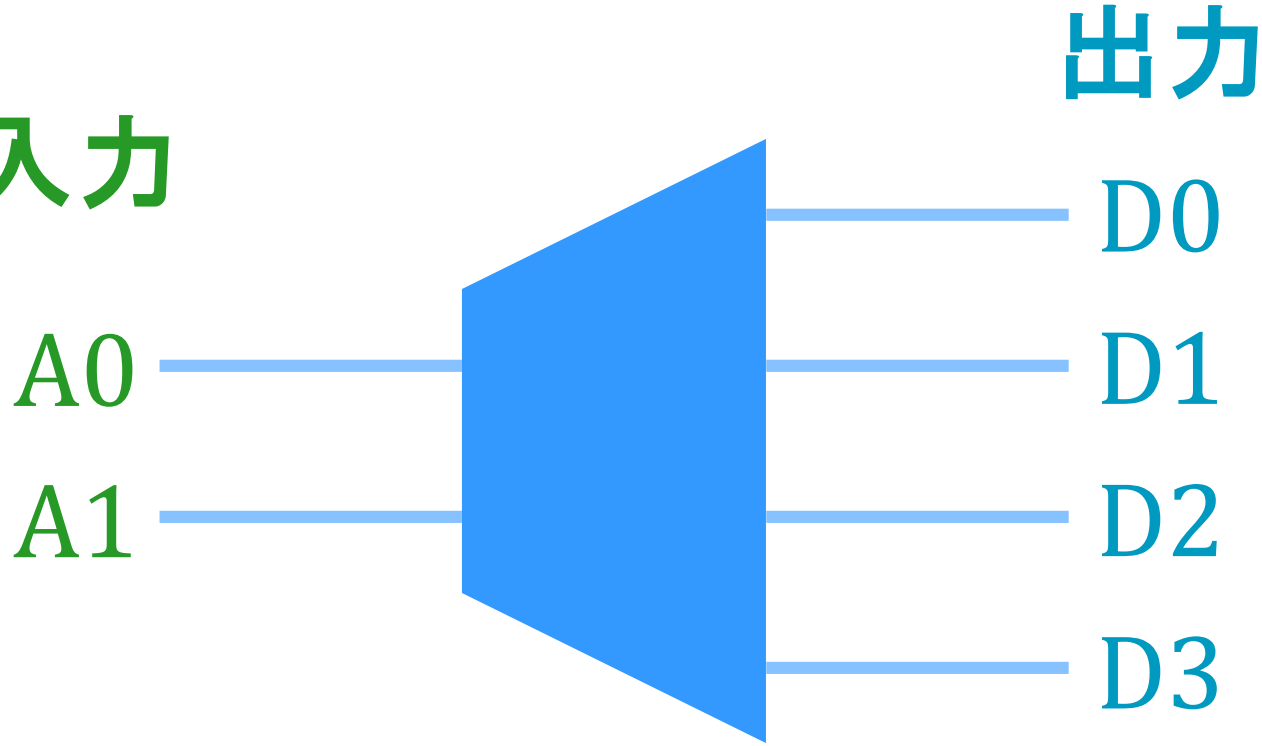
A	B	Z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

論理式

$$C = \overline{A} \cdot B \cdot Z + A \cdot \overline{B} \cdot Z + A \cdot B \cdot \overline{Z} + A \cdot B \cdot Z$$
$$= A \cdot B + B \cdot Z + A \cdot Z$$

$$S = \overline{A} \cdot \overline{B} \cdot Z + \overline{A} \cdot B \cdot \overline{Z} + A \cdot \overline{B} \cdot \overline{Z} + A \cdot B \cdot Z$$

2入力4出力デコーダ



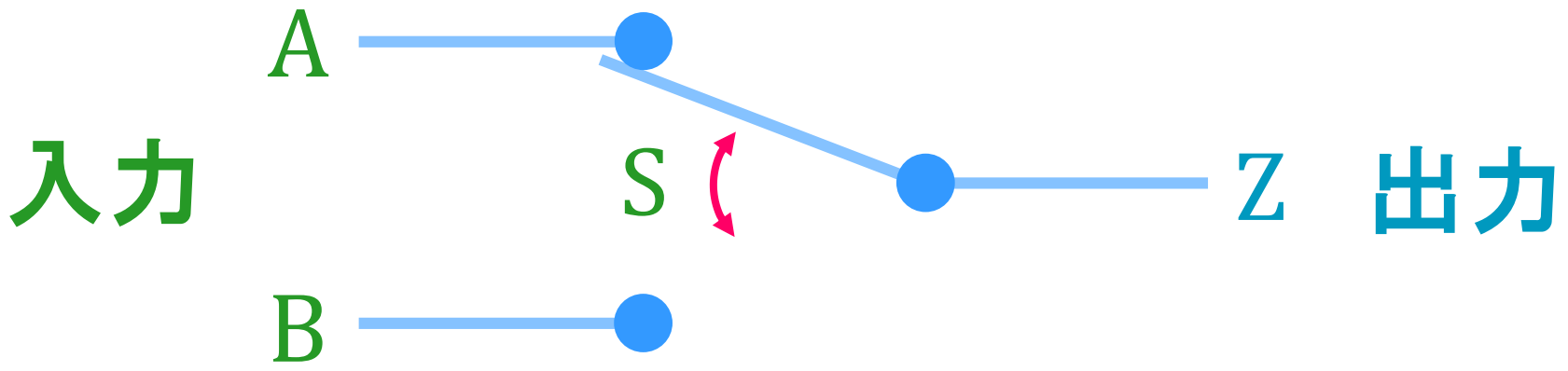
符号化されたデータ

復号

元のデータ

2入力1出力マルチプレクサ

イメージ図



S = 0 のとき Z = A

S = 1 のとき Z = B

真理値表と論理式

2入力4出力デコーダ

A0	A1	D0	D1	D2	D3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

$D0 = \overline{A0} \cdot \overline{A1}$

$D1 = \overline{A0} \cdot A1$

$D2 = A0 \cdot \overline{A1}$

$D3 = A0 \cdot A1$

2入力1出力マルチプレクサ

S	A	B	Z
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

$Z = \overline{S} \cdot A + S \cdot B$

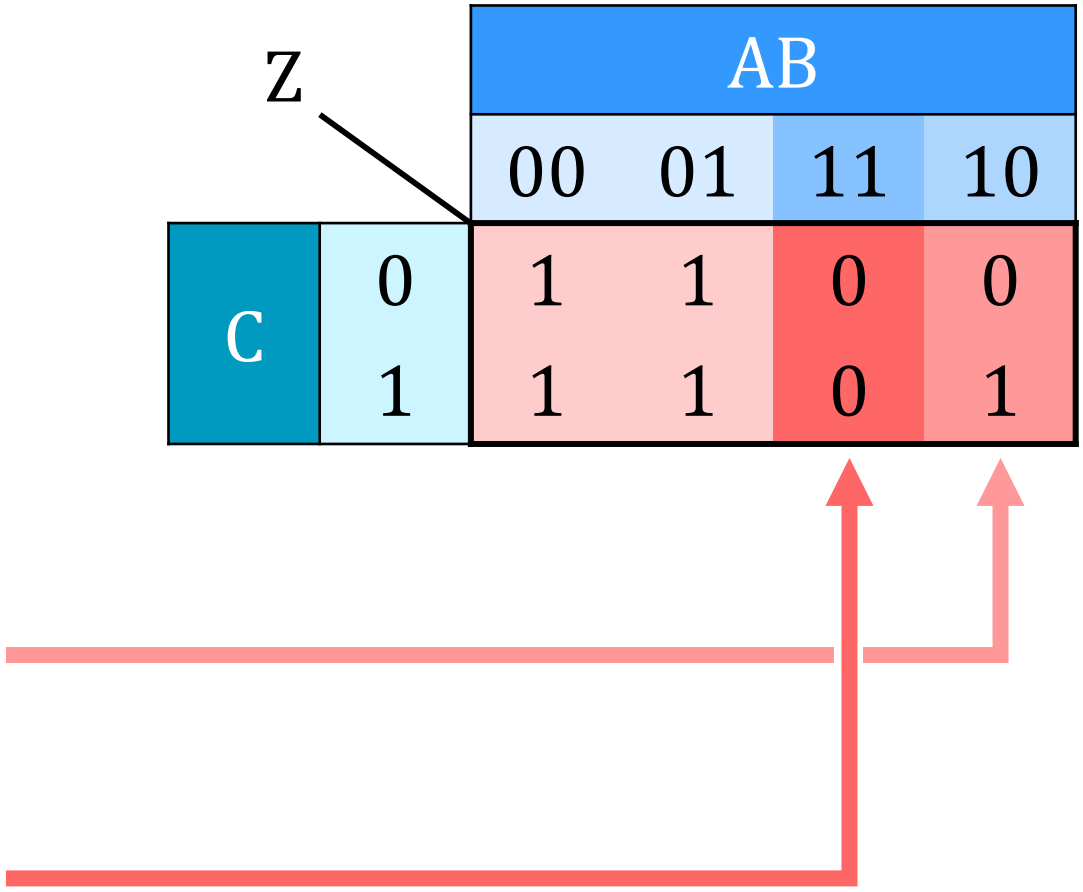
カルノー図

重要

真理値表

A	B	C	Z
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

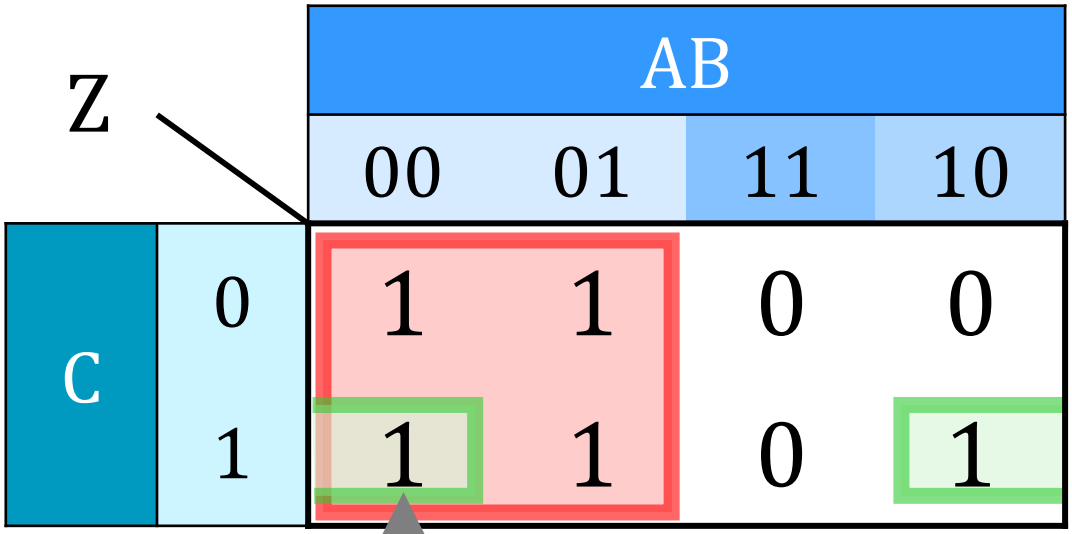
カルノー図



カルノー図による簡単化①

重要

- 「1」のみを四角形で囲む。
- 四角形の幅は、1、2、4のいずれかにする。
- 可能限り大きな四角形で囲む。
- 四角形の数を最小にする。



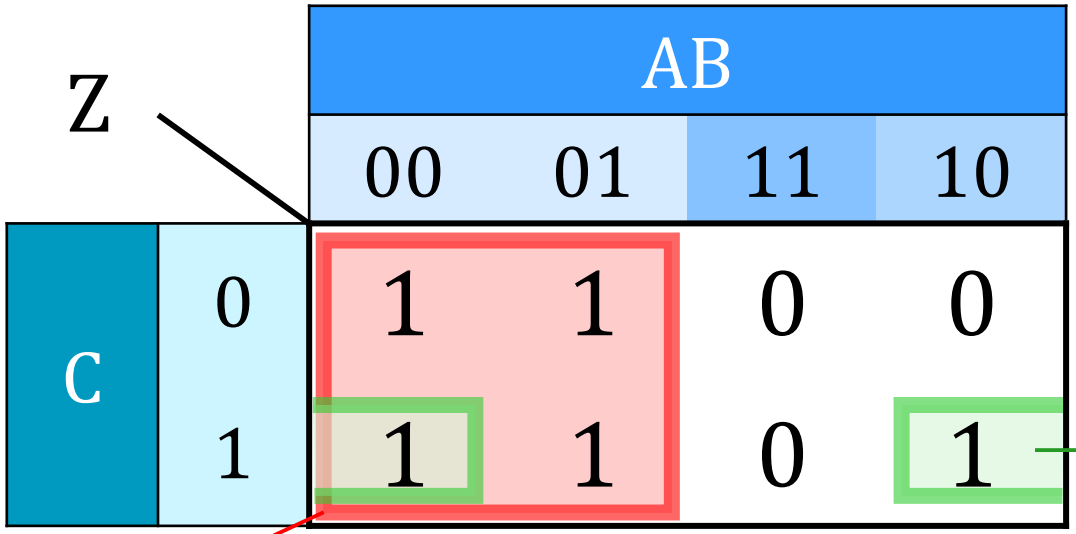
四角形が重なっても良い。

上下、左右は繋がっていると考える。

カルノー図による簡単化②

重要

❶ 各四角形において、入力値が 0 または 1 の片方しか含まれていない変数で、論理積項を表す。
❷ すべての論理積項の論理和を求める。

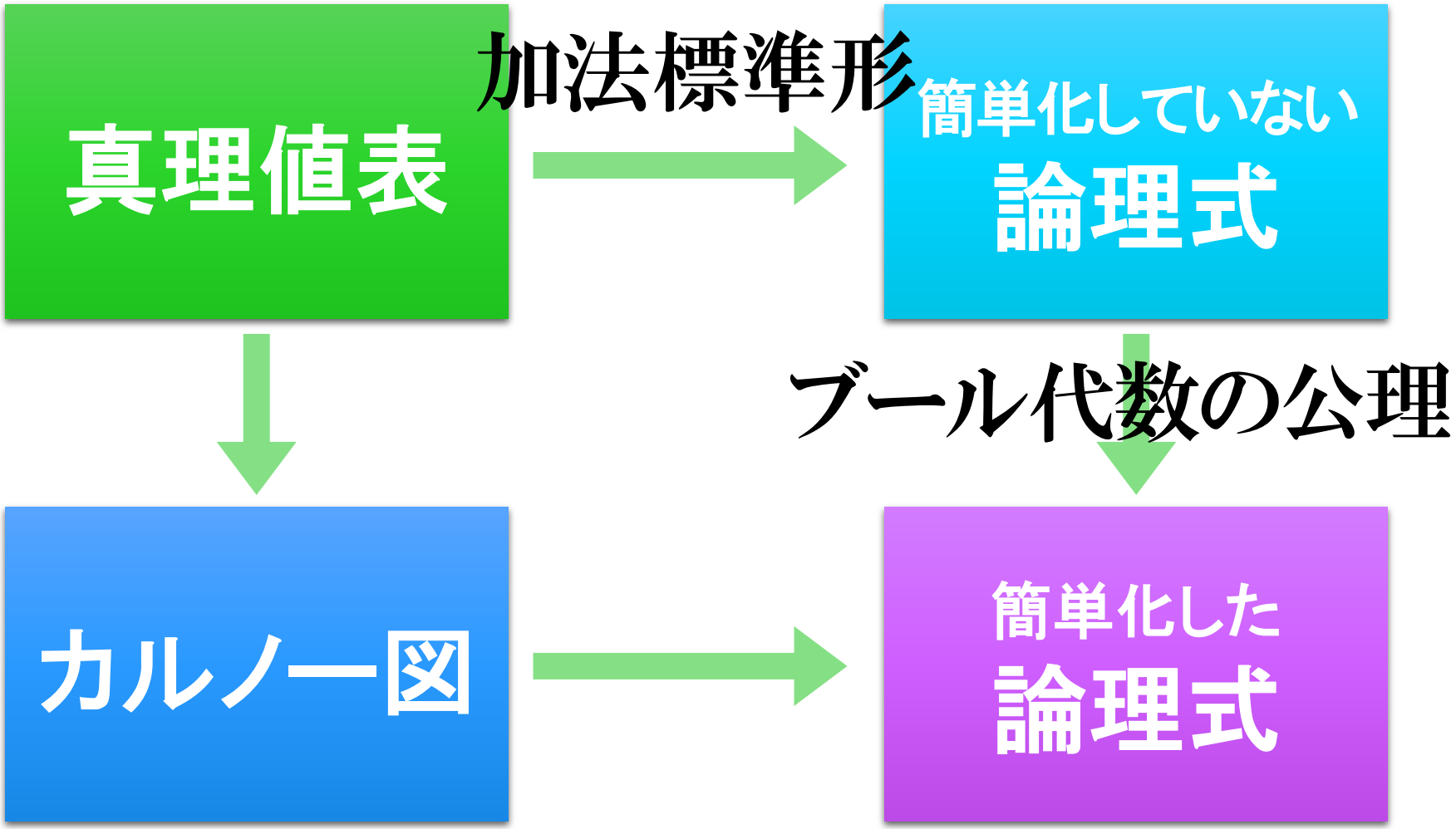


A = 0
B = 0,1
C = 0,1
→ $\overline{A}$

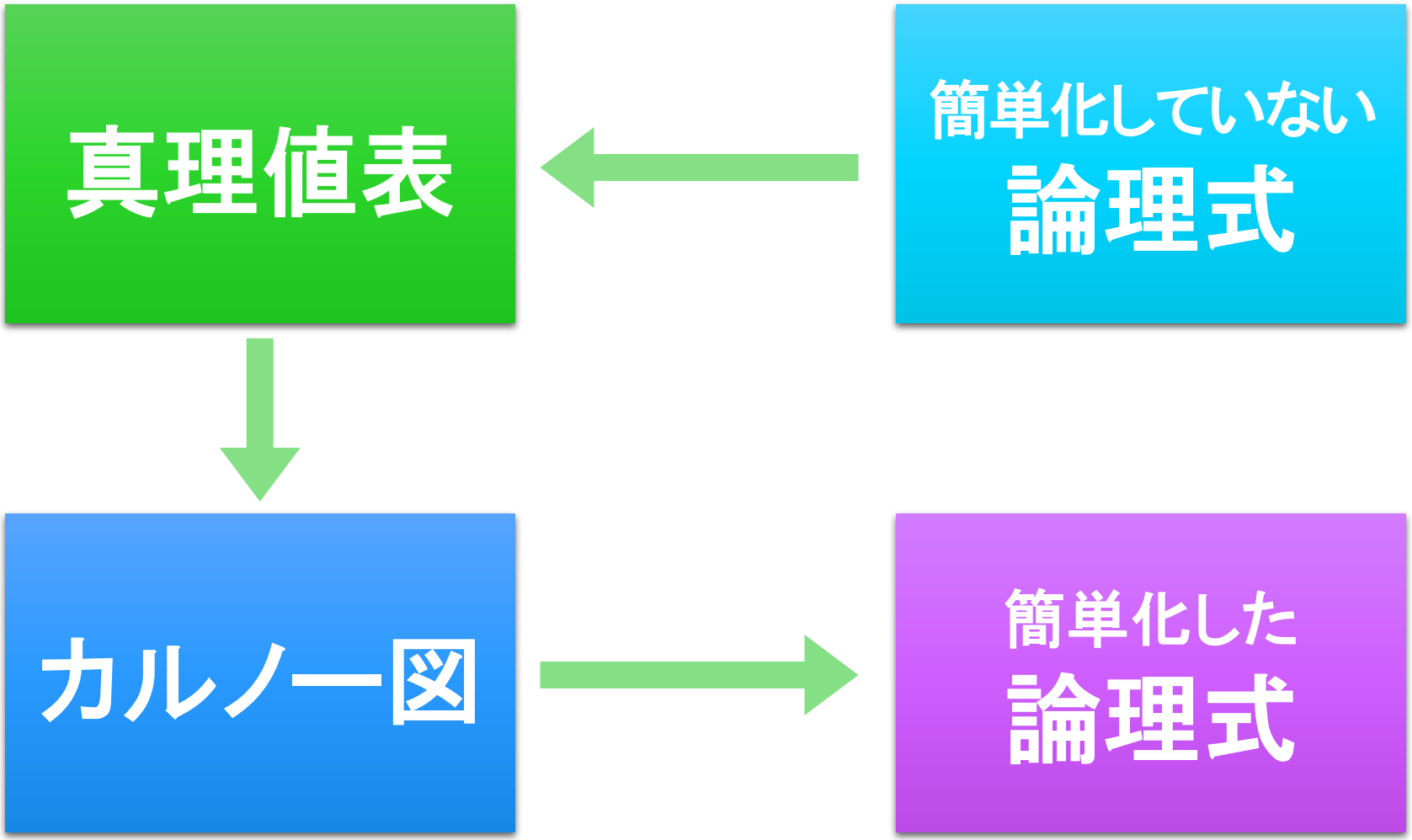
A = 0,1
B = 0
C = 1
→ $\overline{B} \cdot C$

$$Z = \overline{A} + \overline{B} \cdot C$$

真理値表から簡単化した論理式をつくる



カルノー図による論理式の簡単化



課題 2

- 教科書52ページ 演習問題1, 2, 3, 5, 6
- これまでの講義についての感想

  どのような内容（感想・要望）でも良い。

提出日時：12月17日（月） 講義開始前

中間試験　12月17日

論理回路の種類

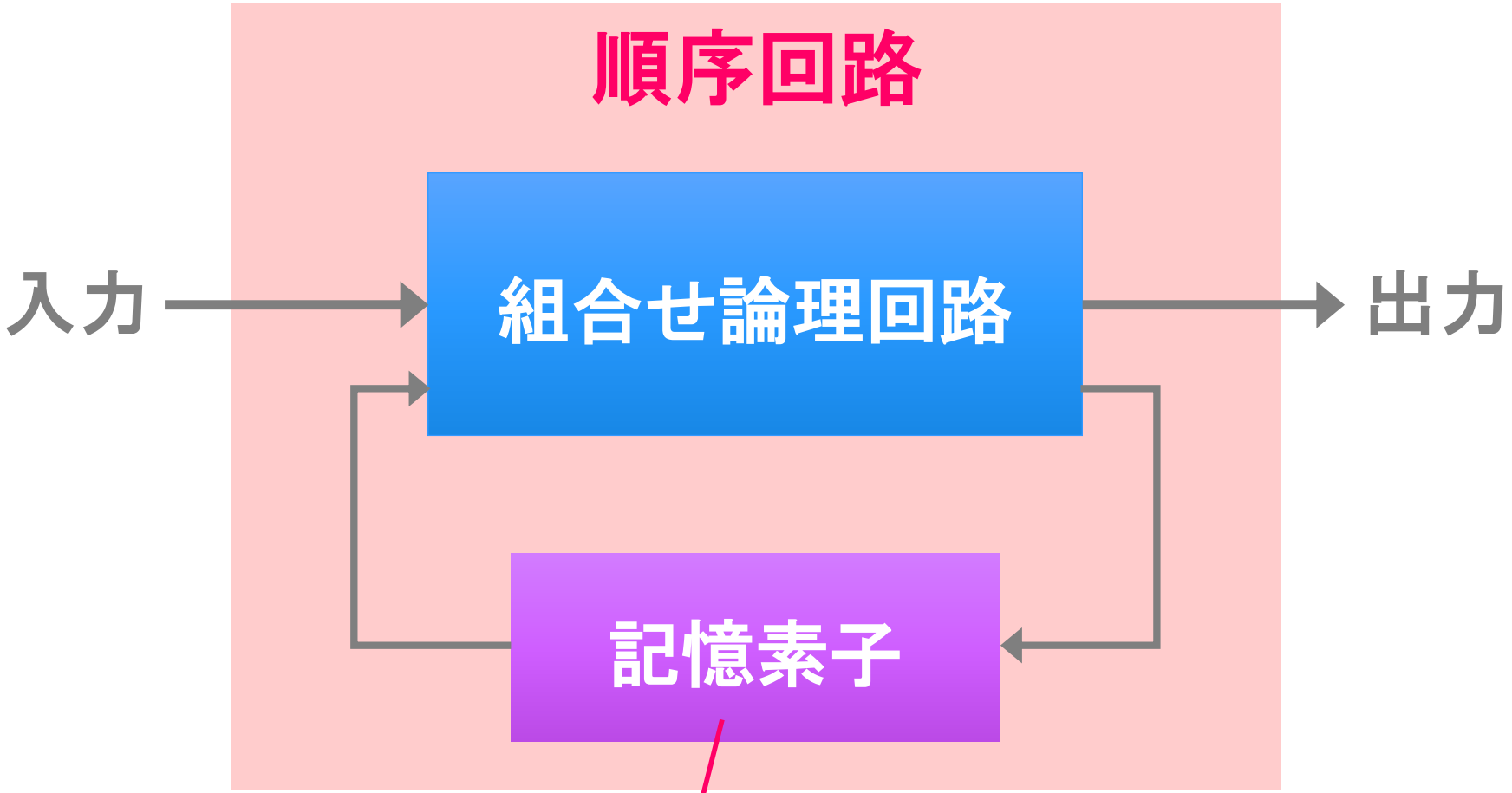
✤ 組合せ論理回路

現在の入力値によって出力値が定まる。

✤ 順序回路

現在の入力値と過去の入力値によって
出力値が定まる。

順序回路



フリップフロップ回路

1bitの値を保持する回路

RSフリップフロップ

重要

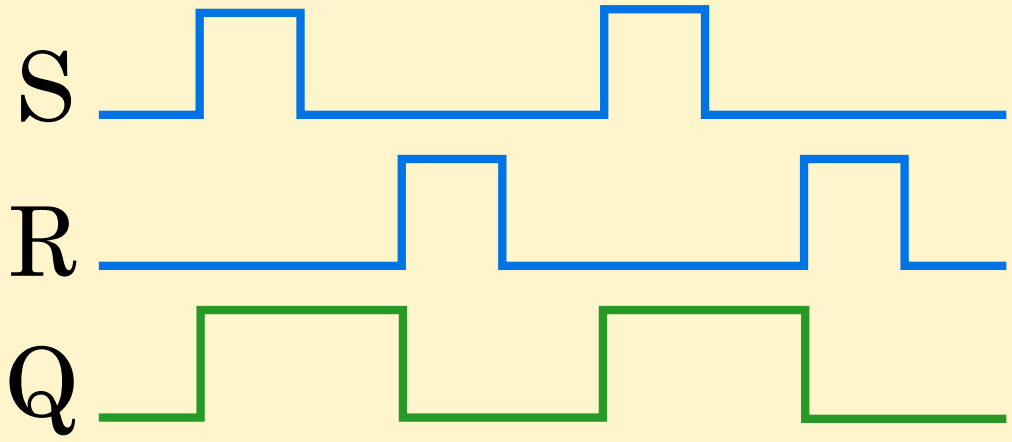
入力信号

R リセット　S セット

出力信号

$Q_{(n)}$ 時刻 n のときの出力

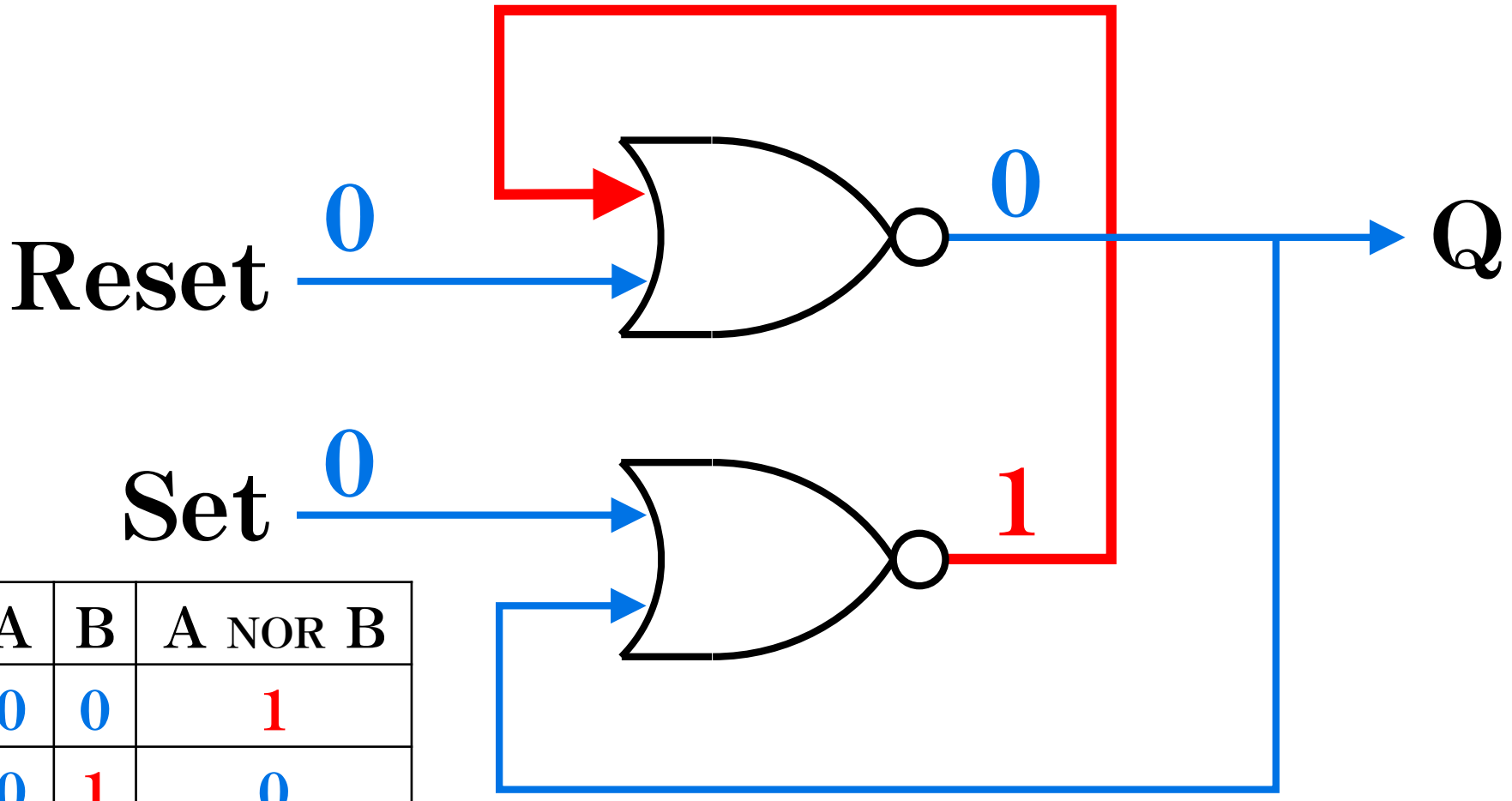
タイミングチャート



動作表

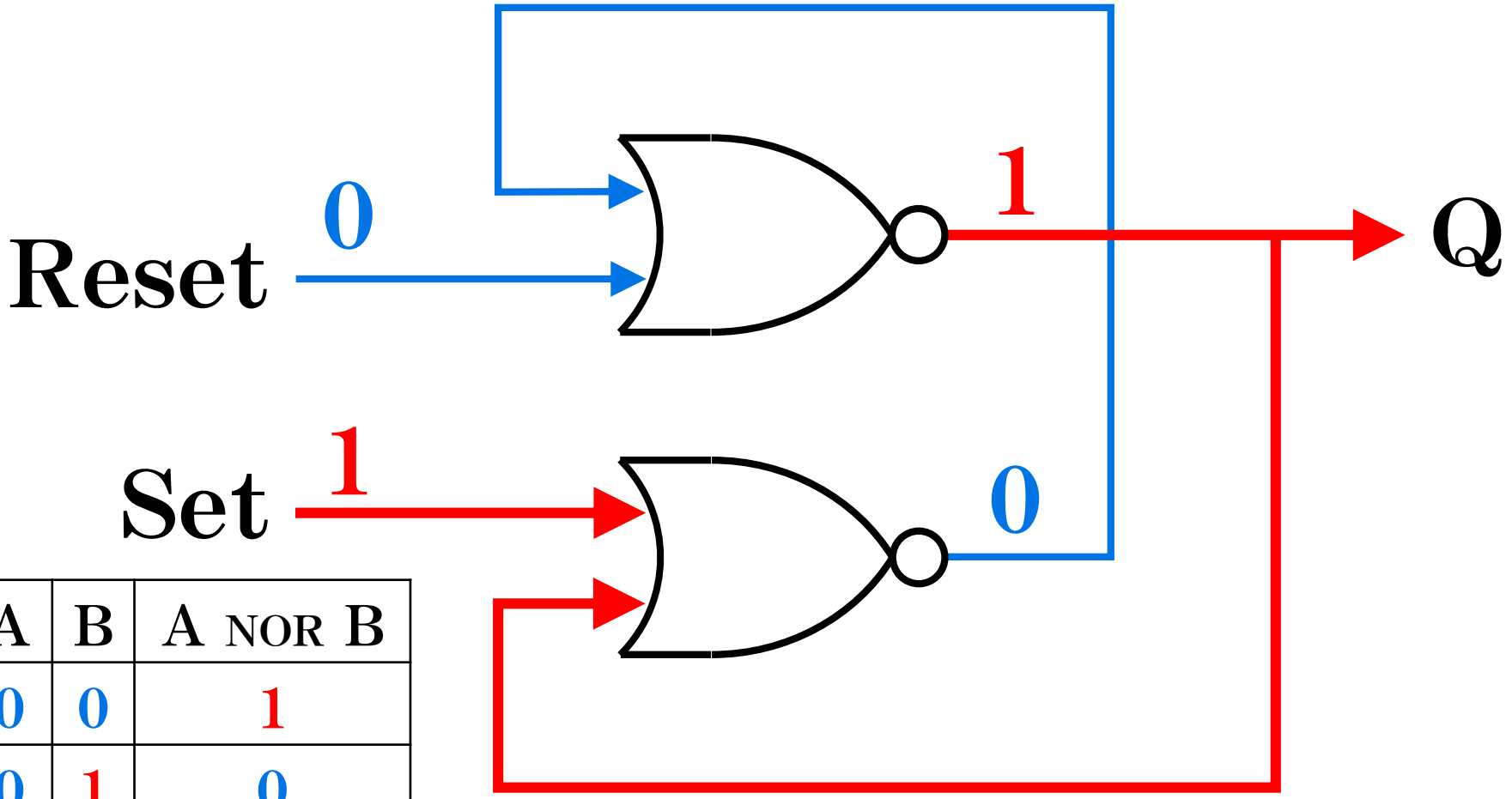
R	S	$Q_{(n)}$
0	0	$Q_{(n-1)}$
0	1	1
1	0	0
1	1	不定

フリップフロップの仕組み



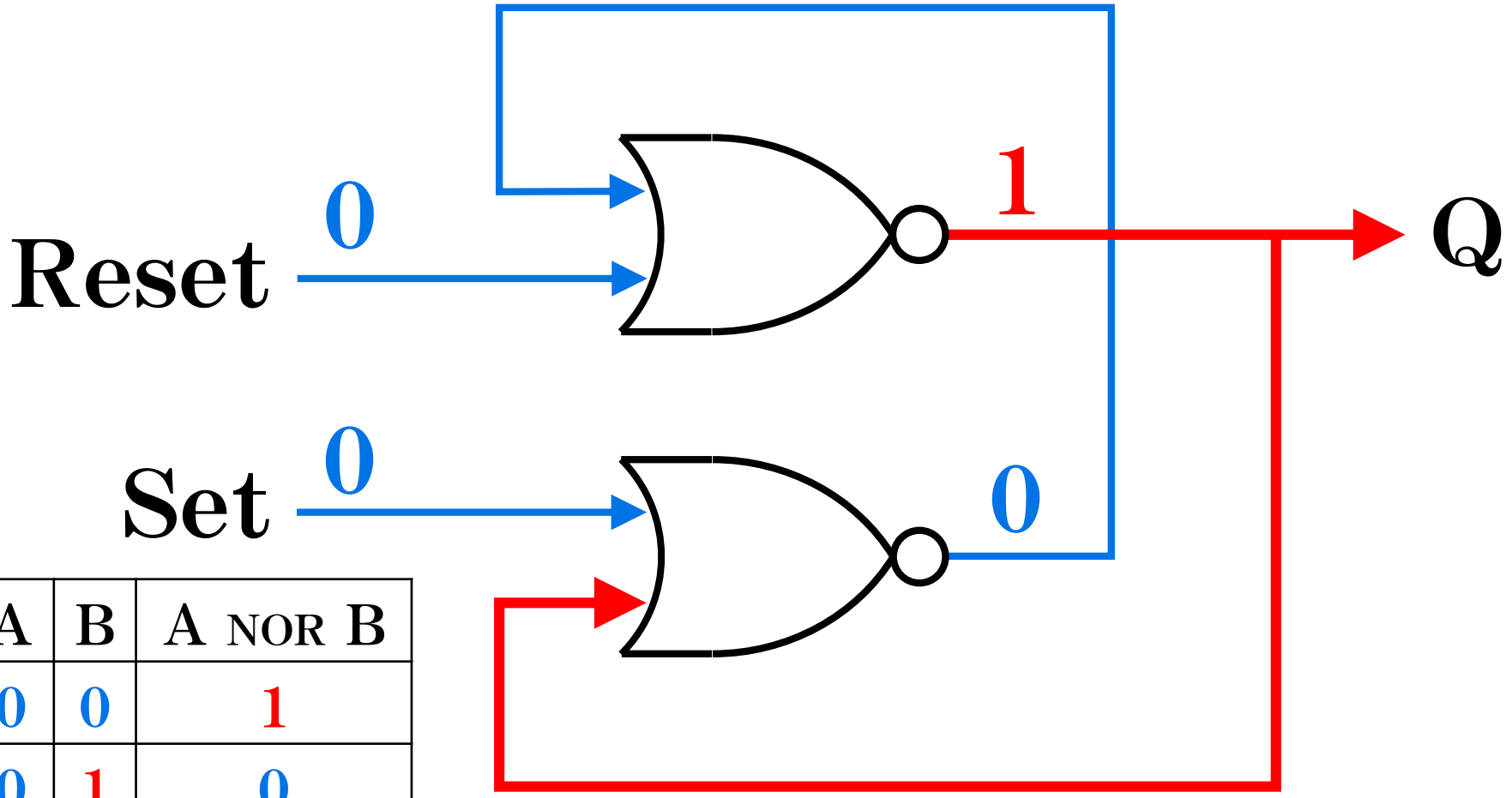
A	B	A NOR B
0	0	1
0	1	0
1	0	0
1	1	0

フリップフロップの仕組み



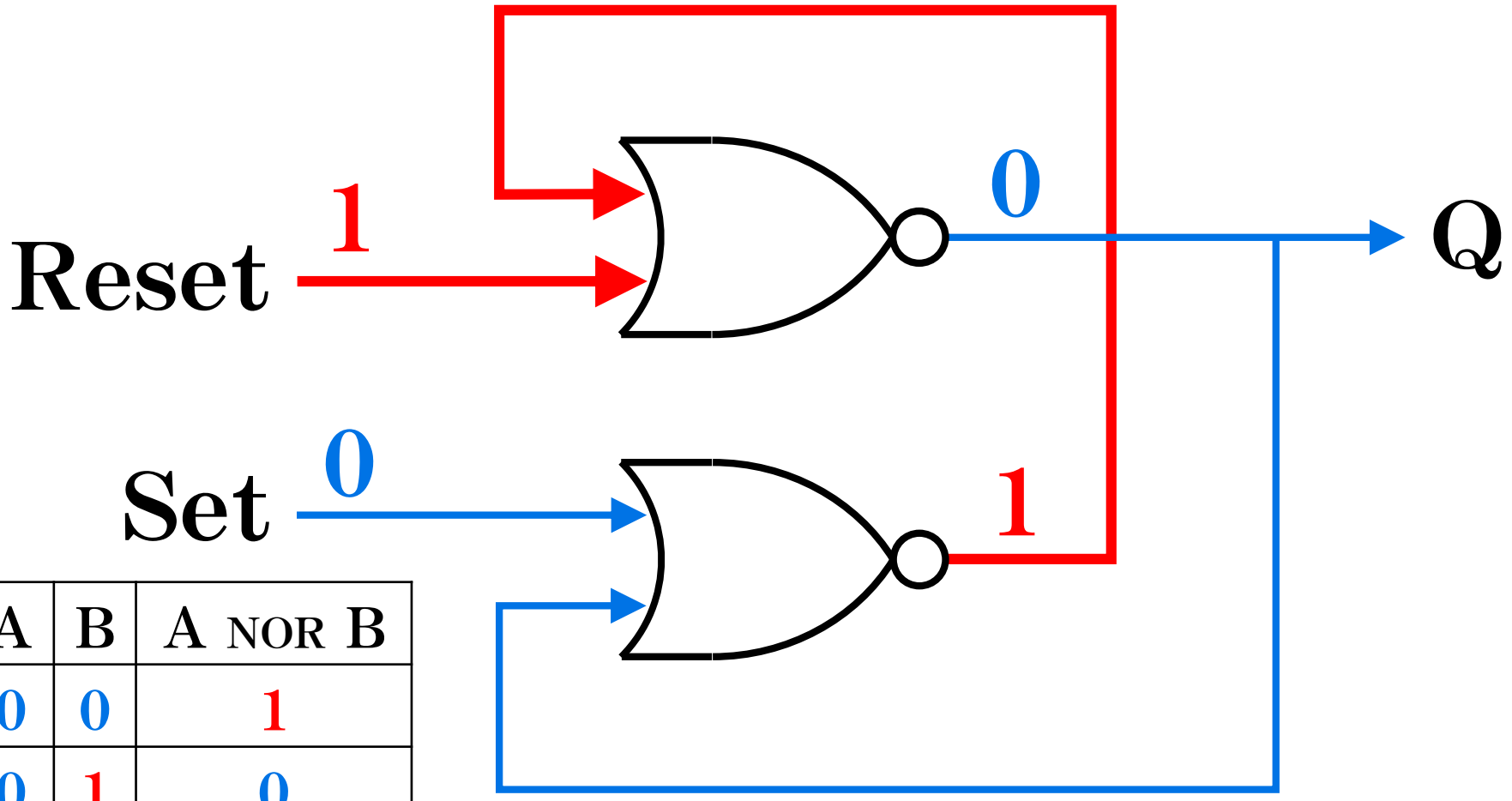
A	B	A NOR B
0	0	1
0	1	0
1	0	0
1	1	0

フリップフロップの仕組み



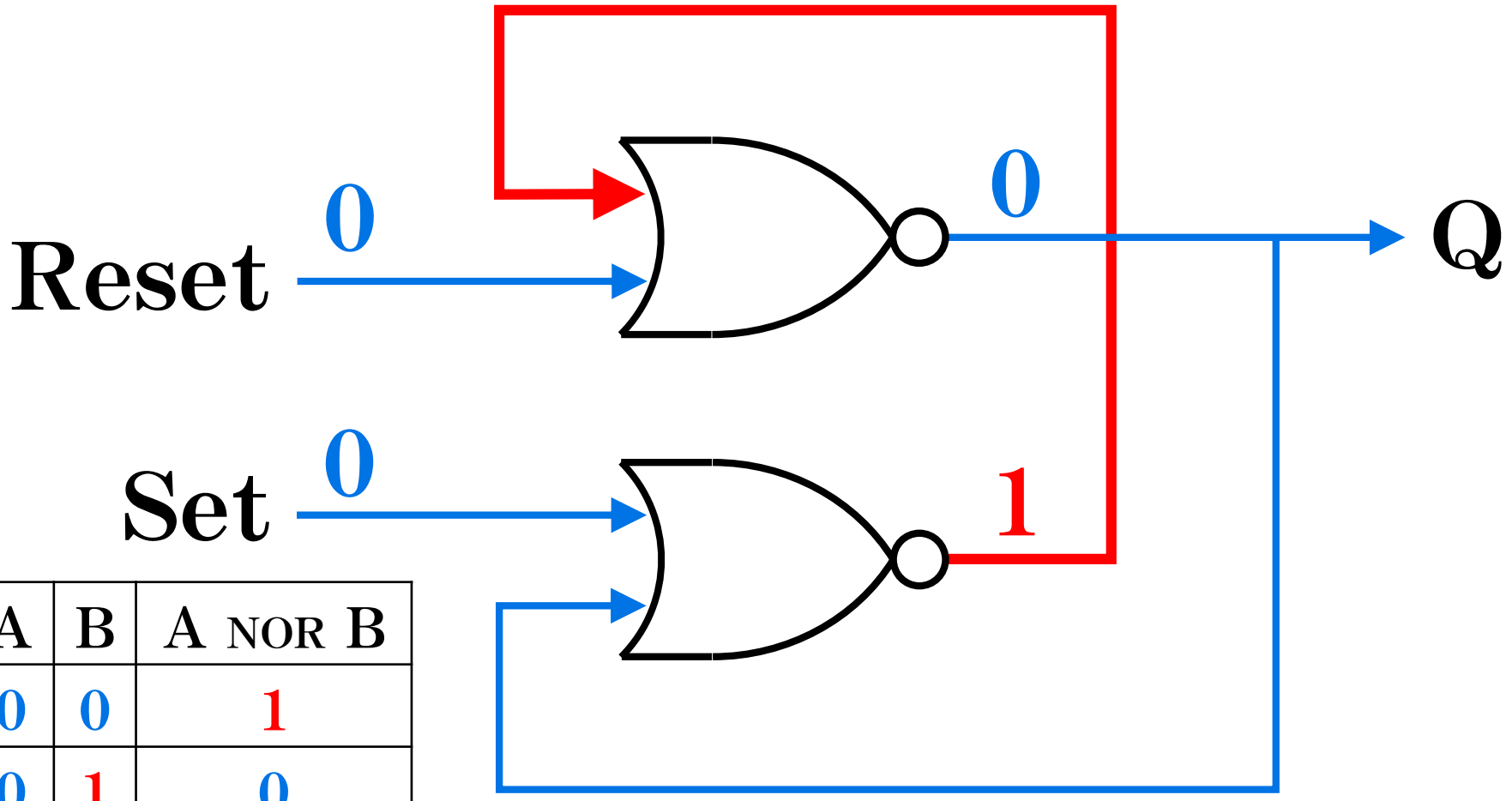
A	B	A NOR B
0	0	1
0	1	0
1	0	0
1	1	0

フリップフロップの仕組み



A	B	A NOR B
0	0	1
0	1	0
1	0	0
1	1	0

フリップフロップの仕組み



A	B	A NOR B
0	0	1
0	1	0
1	0	0
1	1	0

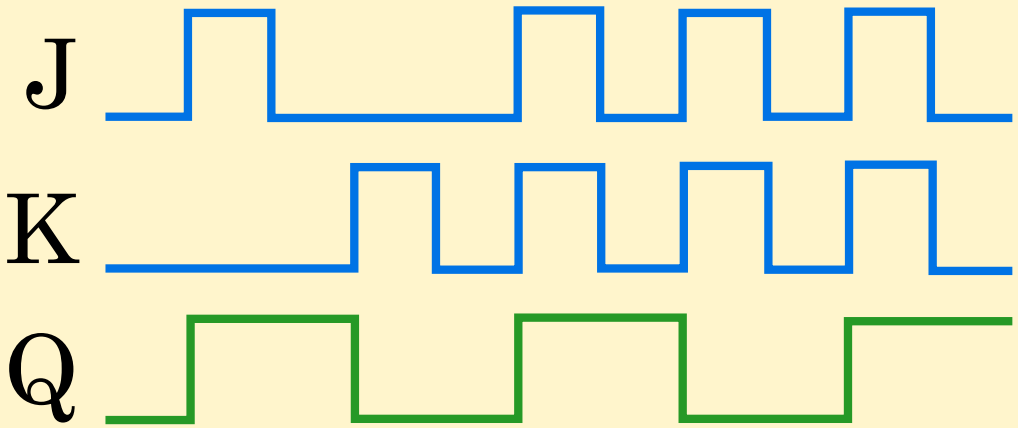
JKフリップフロップ

重要

入力信号

J セット　K リセット

タイミングチャート



動作表

J	K	$Q_{(n)}$
0	0	$Q_{(n-1)}$
0	1	0
1	0	1
1	1	$\overline{Q_{(n-1)}}$

同期式フリップフロップ

重要

クロック信号

複数の電子回路の間で信号の送受信のタイミングを合わせる（同期をとる）ために用いる一定周期の信号

同期式フリップフロップ

クロック信号の立上り（または立下り）のときの入力値で動作するフリップフロップ

その他のフリップフロップ

Dフリップフロップ

入力信号Ｄを1クロック周期保持する。

Tフリップフロップ

トリガ信号Ｔが入力されるたびに出力を反転する。

フリップフロップの応用

CPUの中の記憶装置

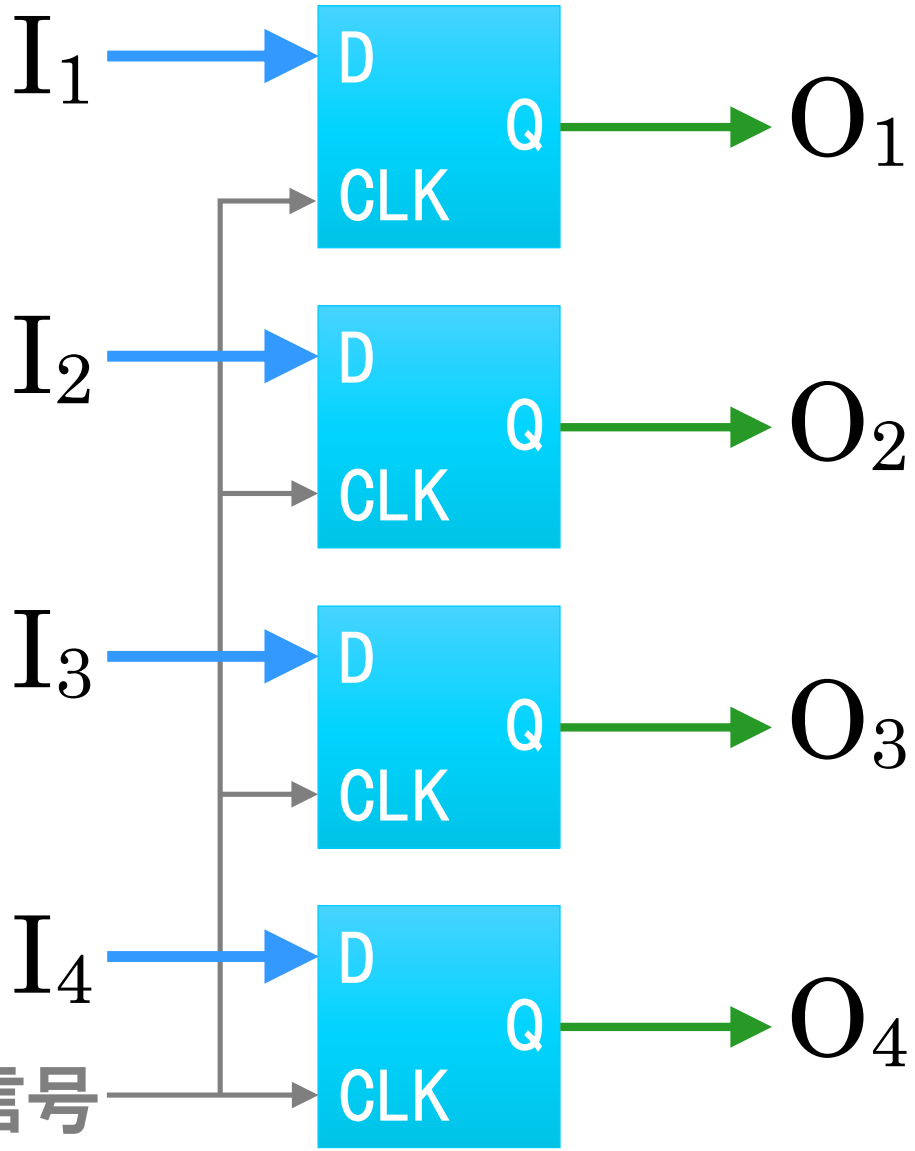
- レジスタ
- シフトレジスタ
- カウンタ
- 基本記憶素子（キャッシュメモリ）

レジスタ

重要

演算に用いる値や命令コードなどを一時的に記憶する装置

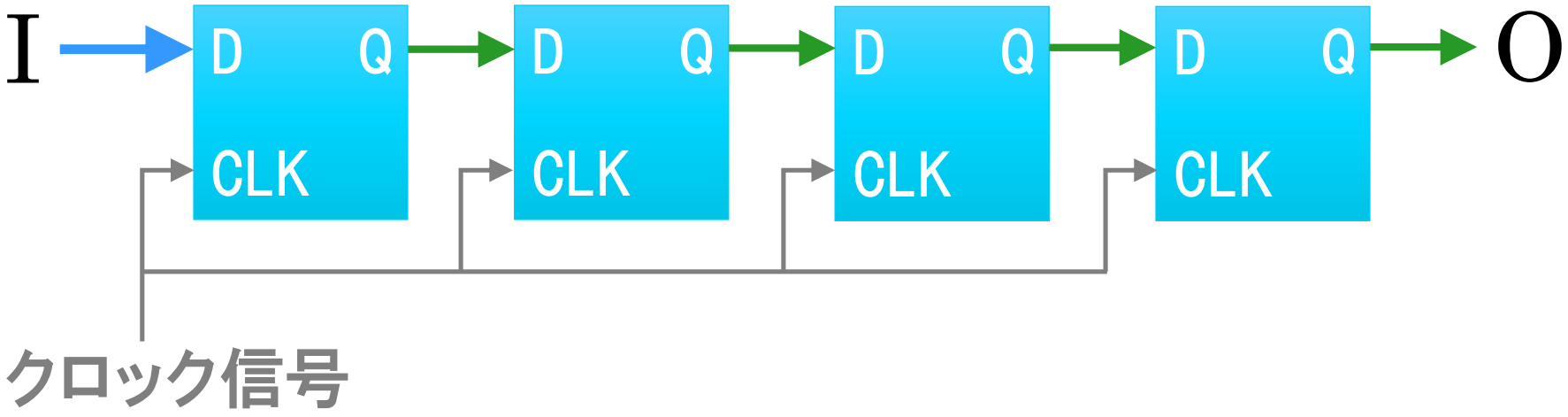
4bitの値 $I_4I_3I_2I_1$ を記憶するレジスタ



シフトレジスタ

クロック信号が入力されるたびに、数値が1bitシフトするレジスタ

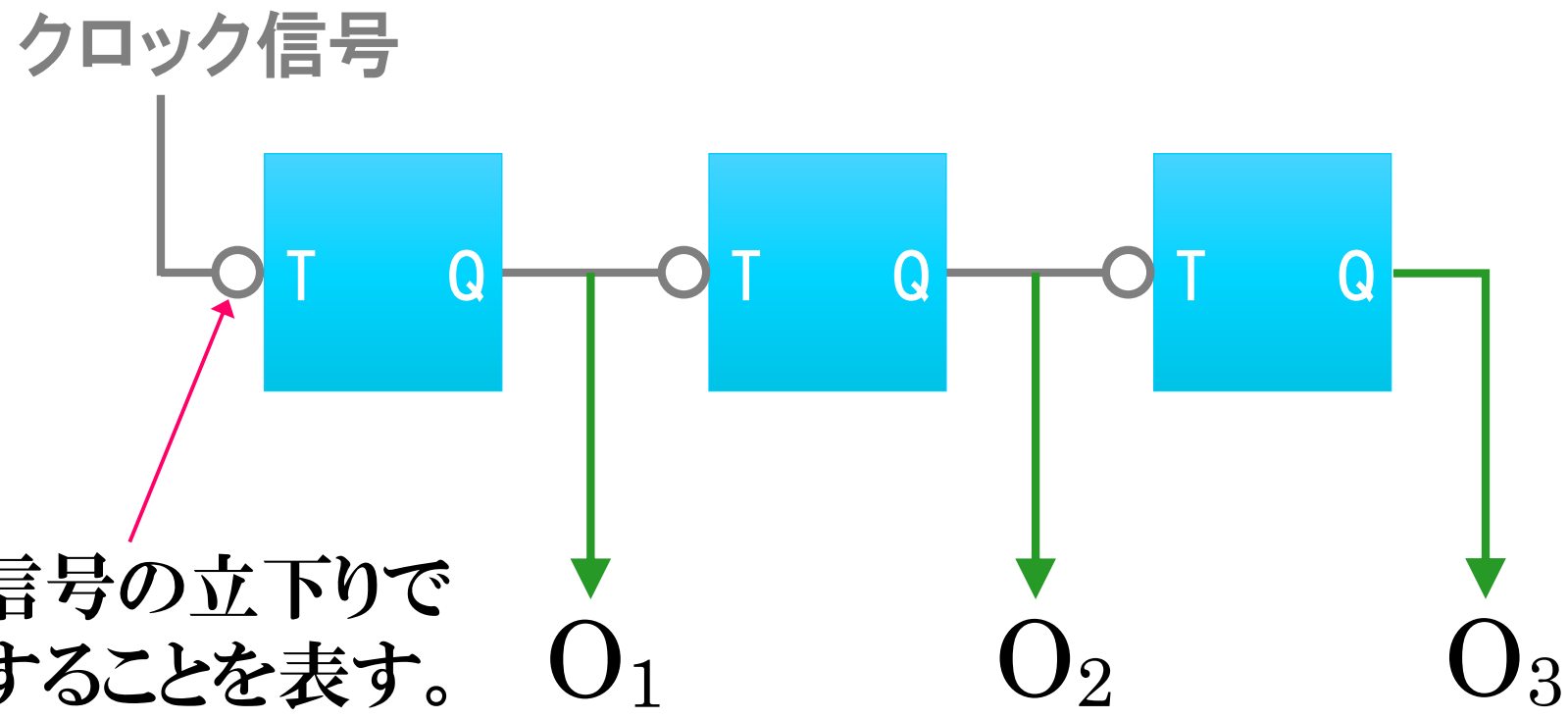
データを1bitずつ送信する(シリアル通信)。



カウンタ

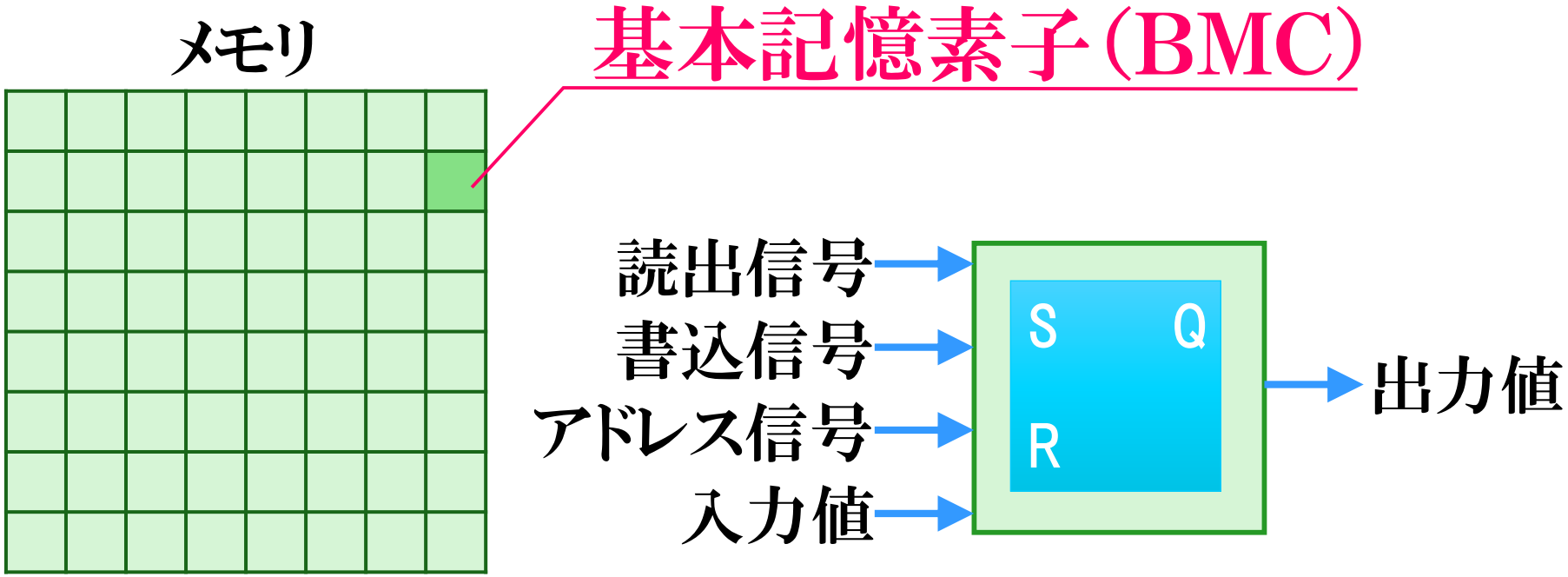
重要

入力信号のパルス数をかぞえる機能を持つレジスタ



基本記憶素子

アレイ状のメモリを構成する1bitの記憶素子



CPUのキャッシュメモリのBMCは、フリップフロップを用いている。