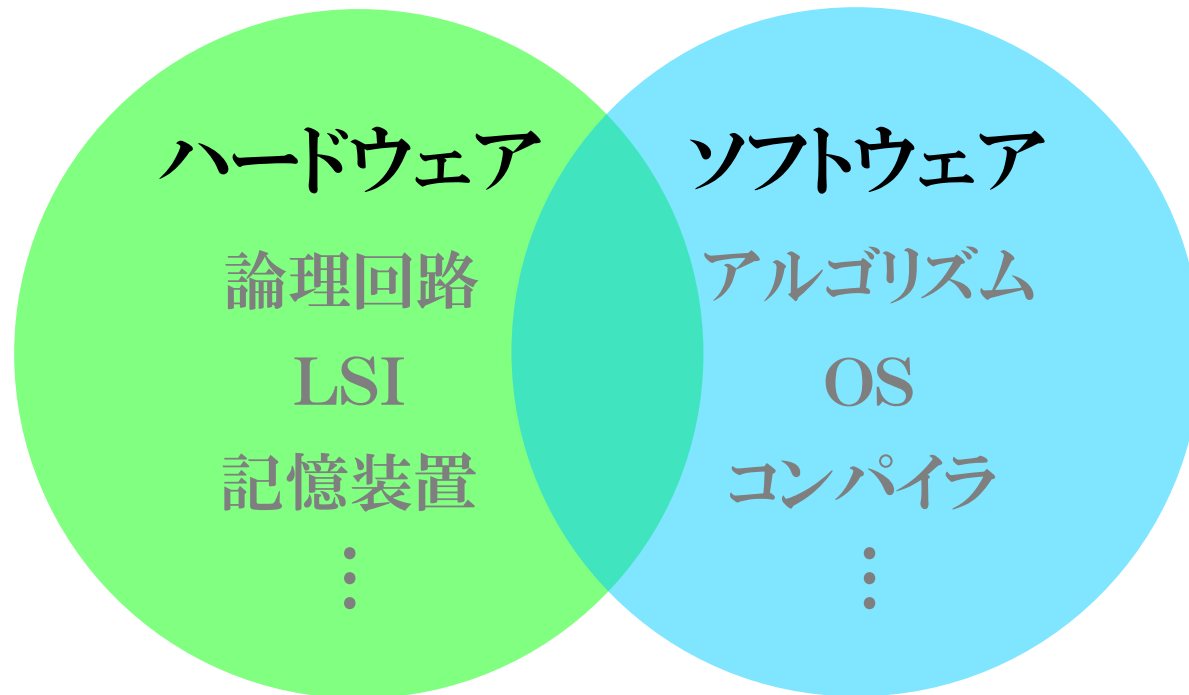


コンピュータ工学 I

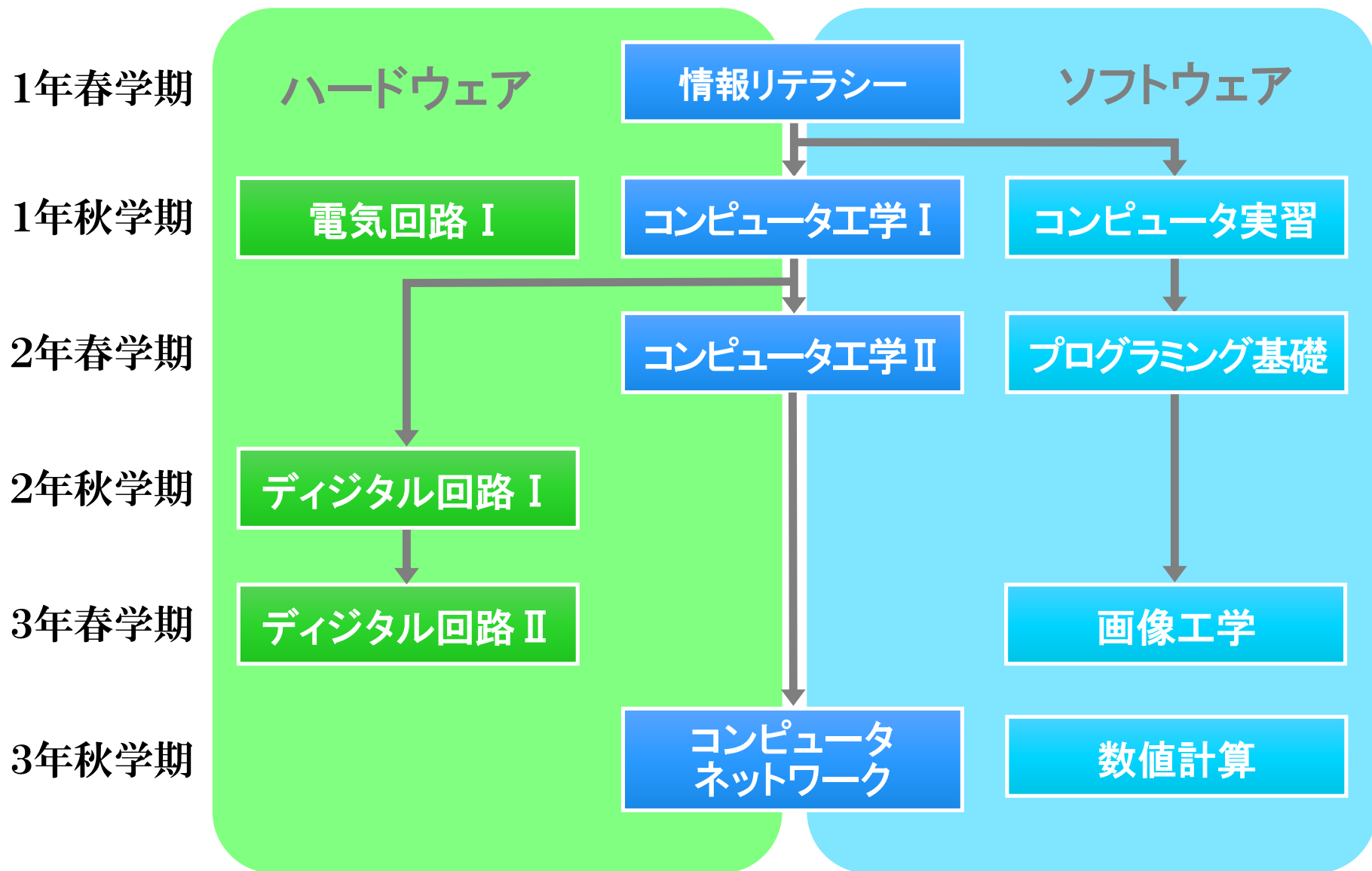
Rev. 2017.12.04

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側面から研究する学問。



カリキュラム体系



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

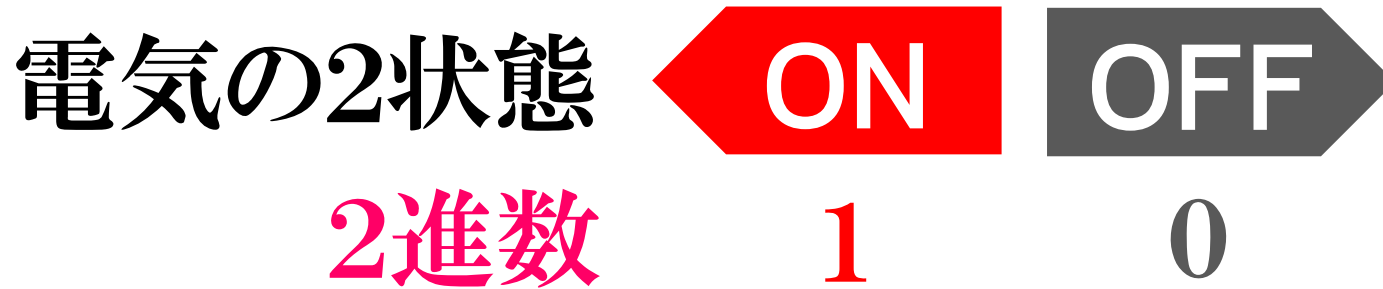
- ① 数と文字の表現方法
- ② 論理回路の設計
- ③ CPUの構成要素
- ④ アセンブリ言語とCPUの動作

数と文字の表現方法

✦ 内容

- ① 正の整数
- ② 演算
- ③ 負の整数
- ④ 実数
- ⑤ 文字

データの記憶・伝達



101110001110011111010100001101

2進数に変換

数値データ

2015 2.718281828
-322 6.0221×10^{23}

数値化

音声データ

画像データ

文章データ

正の整数

❖ 数の表記

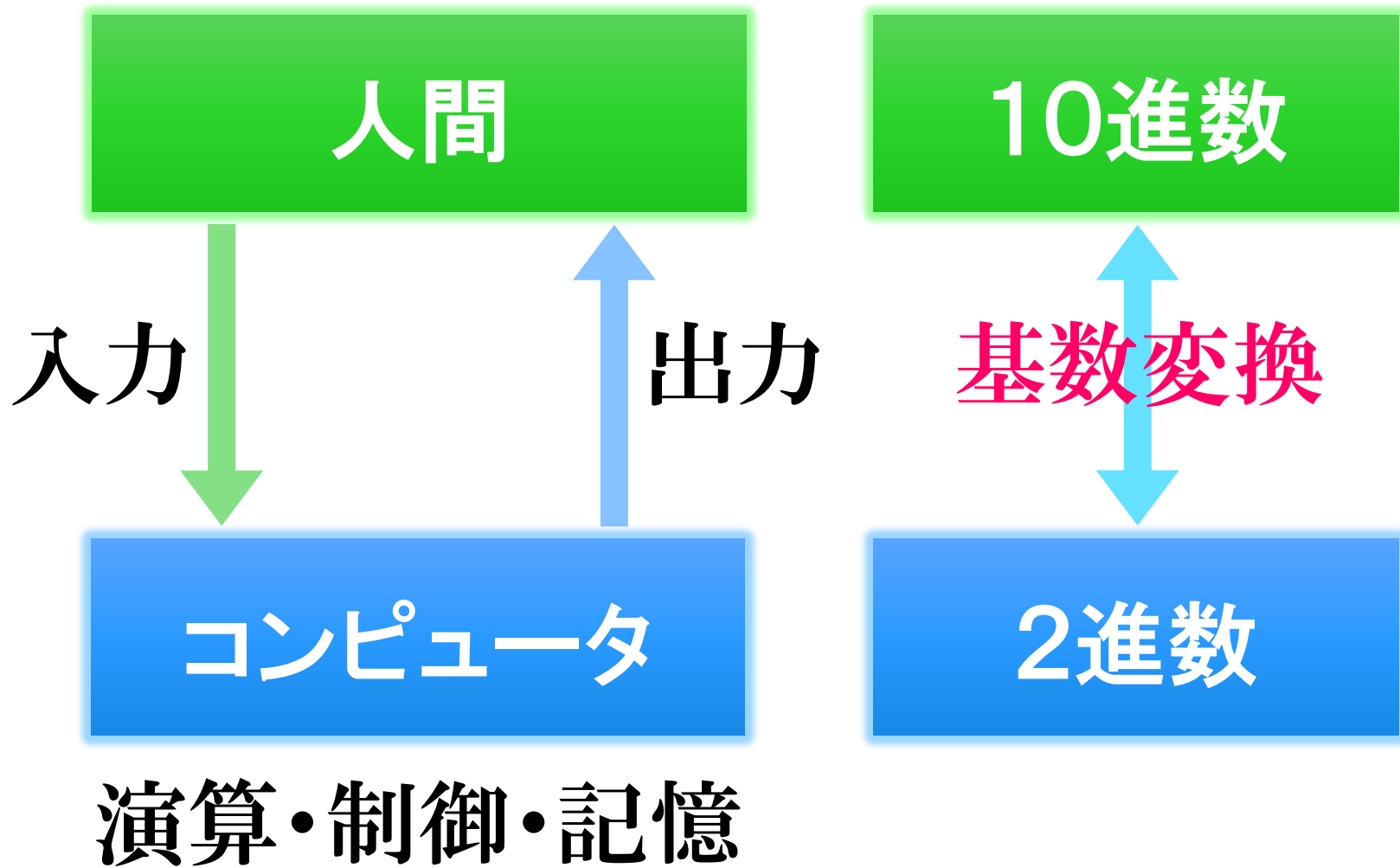
$N_{(p)}$: p 進数の数値 N

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



p進数整数→10進数

10進数を $N_{(10)}$ 、

p 進数整数を $d_n d_{n-1} \cdots d_1 d_0 (p)$ とする。

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

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

10進数を $N_{(10)}$ 、 p 進数を

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

とする。 $(n > 0 > m)$

重要!

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

10進数整数 \rightarrow p進数

重要!

$$p \) \ \underline{N_{(10)}} \quad \text{余り}$$

$$p \) \ \underline{q_0} \quad \cdots \ d_0$$

$$p \) \ \underline{q_1} \quad \cdots \ d_1$$

\vdots

$$p \) \ \underline{q_{n-1}} \quad \cdots \ d_{n-1}$$

$$0 \quad \cdots \ d_n = q_{n-1}$$

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

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, \dots, d_n が順番に求まる。

10進数小数部→p進数

重要!

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

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

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

⋮

$b_n = 0$ になるまで続ける。

$$N_{(10)} = 0.a_0a_1 \cdots a_n (p)$$

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

a_i 整数部

b_i 小数部

とする。

基数変換まとめ

❖ p 進数 \rightarrow 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 \rightarrow p 進数

整数部を p で割っていき、余りを右から順
に並べる。

小数部に p を掛けていき、整数値を左か
ら順に並べる。

16進数

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。

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

$$\begin{array}{r} 0 \\ + 0 \\ \hline 0 \end{array}$$

$$\begin{array}{r} 1 \\ + 0 \\ \hline 1 \end{array}$$

$$\begin{array}{r} 1 \\ + 1 \\ \hline 10 \end{array}$$

$$\begin{array}{r} 1 \\ 1 \\ + 1 \\ \hline 11 \end{array}$$

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

- ❖ 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。

重要!

n bit 左シフト $\rightarrow 2^n$ 倍になる。

n bit 右シフト $\rightarrow \frac{1}{2^n}$ 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。

$$\begin{array}{r} 111 \\ \times 101 \\ \hline 111 \\ + 11100 \\ \hline 100011 \end{array}$$

111を0 bit左シフト
111を2 bit左シフト
答え

負の整数

コンピュータは 0 と 1 しか使えない。



+と-の符号が存在しない

❖ 負の数の表現方法

- ① 符号絶対値法
- ② 1の補数
- ③ 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 \bar{x} とするとき、

$$x + \bar{x} = 111\cdots 1_{(2)}$$

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、

$$x + x' = 100\cdots 0_{(2)}$$

となる。

✦ 求め方

重要!

x の1の補数に1を足す。

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。

$$2 - 5 =$$



$$2 + (-5) =$$

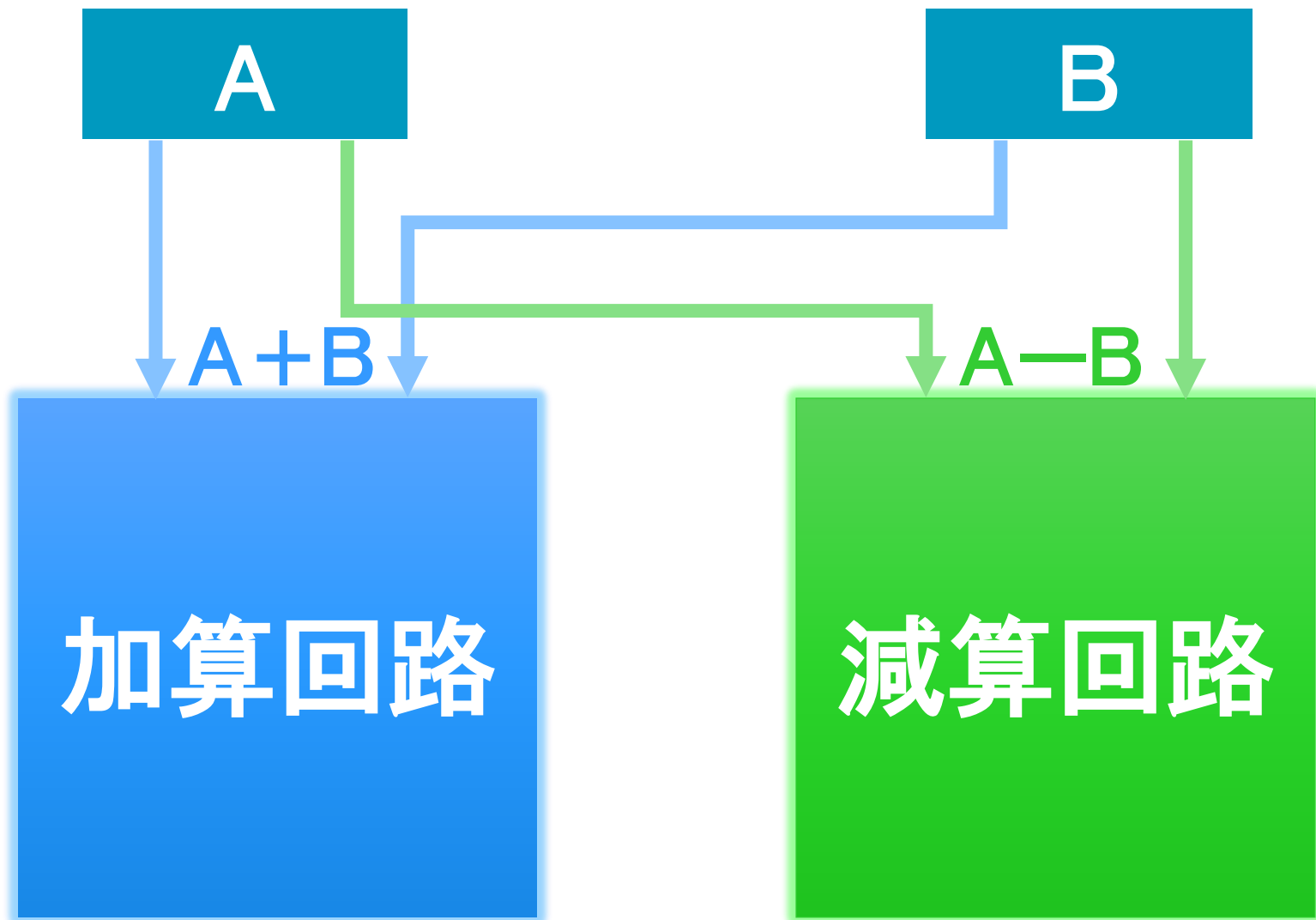
コンピュータ設計の方針

♣ 良いコンピュータとは？

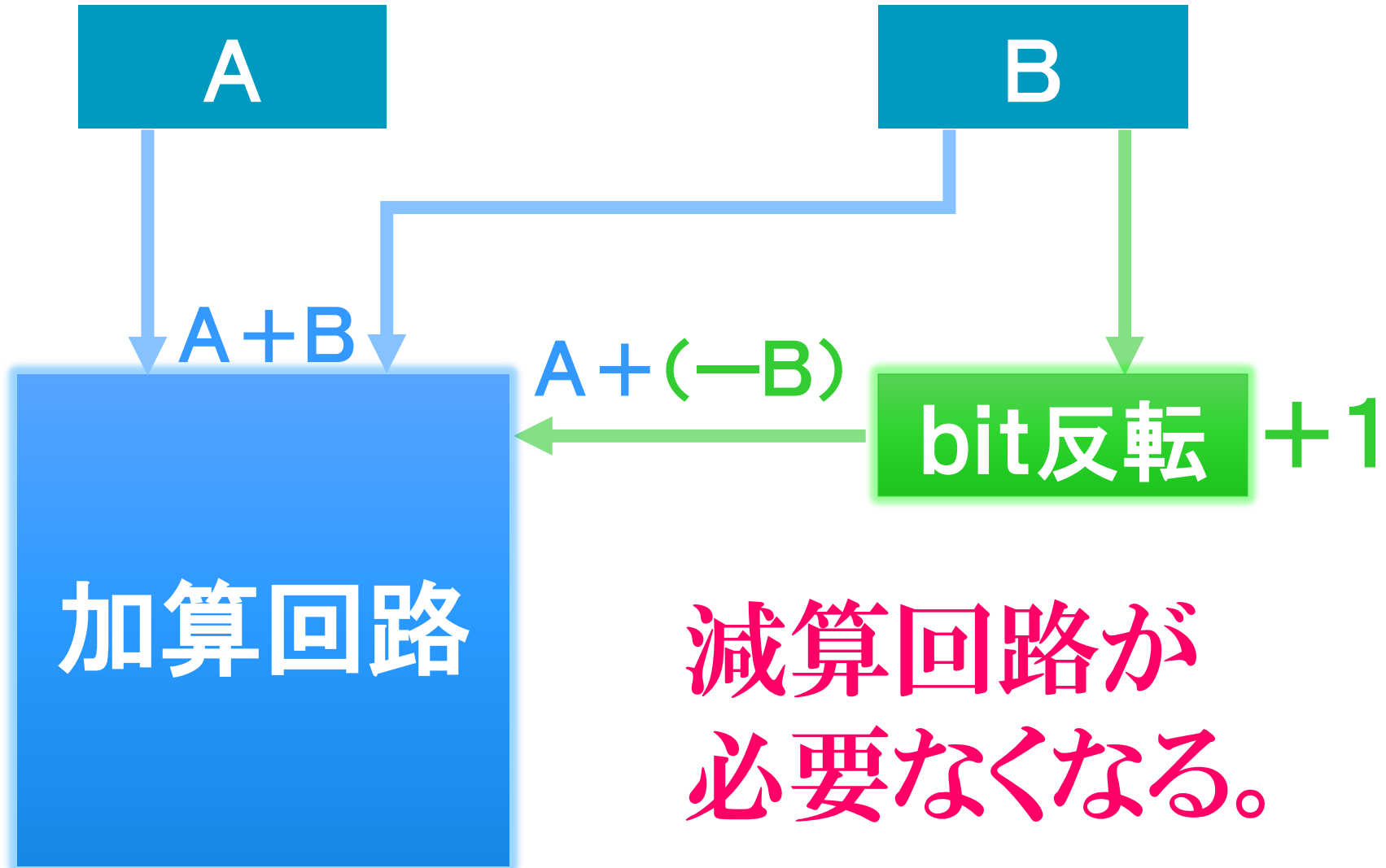
- 高速
- 小型
- 低コスト

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

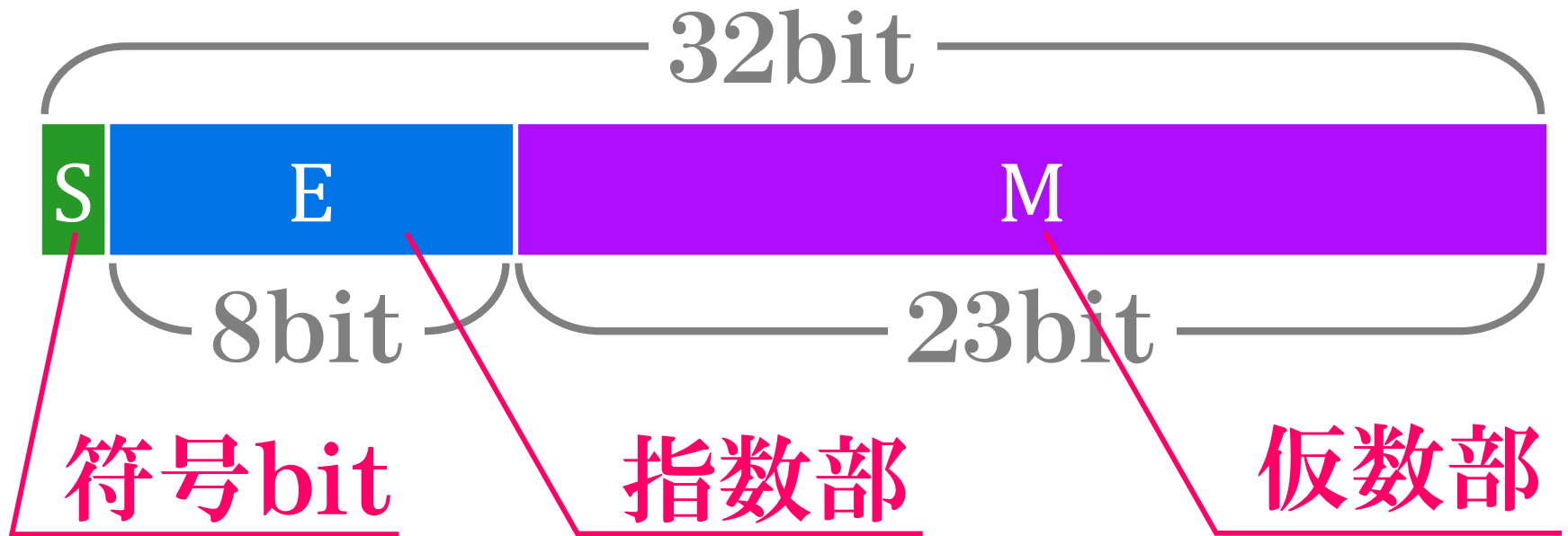
小数点の位置を特定のbitの間に固定する。

整数は、小数部のない固定小数点数

❖ 浮動小数点数

仮数部と指数部に分けて数値を表現する。

单精度浮動小数点数



重要!

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

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	-32768~32767
long (int)	32	-2147483648~2147483647

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

$$\begin{array}{r} 1.1101 \times 2^5 \\ + 1.1000 \times 2^3 \\ \hline \end{array} \quad \rightarrow \quad \begin{array}{r} 1.1101 \times 2^5 \\ + 0.0110 \times 2^5 \\ \hline 10.0011 \times 2^5 \\ \downarrow \\ 1.0001 \times 2^6 \end{array}$$

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

表現できる数値の
範囲が狭い

計算が速い

浮動小数点数

表現できる数値の
範囲が広い

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

上位 4bit

文字コード

41 (16)

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			ア	キ	ヌ	ラ		
8			(8	H	X	h	x			イ	ク	ネ	リ		
9)	9	I	Y	i	y			ウ	ケ	ノ	ル		
A			*	:	J	Z	j	z			エ	コ	ハ	レ		
B			+	;	K	[k	{			オ	サ	ヒ	ロ		
C			,	<	L	¥	l				ヤ	シ	フ	ワ		
D			-	=	M]	m	}			ユ	ス	ヘ	ン		
E			.	>	N	^	n	~			ヨ	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~

C

i

t

y

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

1B284A

制御コードを
埋め込む

16bitコード開始

8bitコード開始

16bit シフトJISコード

16bit

8bit

東京City

938C

8B9E

43

69

74

6F

(16)

未定義

未定義

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	未定義域			ー	タ	ミ	未定義域	
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			ア	キ	ヌ	ラ		
8			(8	H	X	h	x			イ	ク	ネ	リ		
9)	9	I	Y	i	y			ウ	ケ	ノ	ル		
A			*	:	J	Z	j	z			エ	コ	ハ	レ		
B			+	;	K	[k	{			オ	サ	ヒ	ロ		
C			,	<	L	¥	l				ヤ	シ	フ	ワ		
D			-	=	M]	m	}			ユ	ス	ヘ	ン		
E			.	>	N	^	n	~			ヨ	セ	ホ	ゝ		
F			/	?	O	_	o				ッ	ソ	マ	。		

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

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14日(木) 講義開始前

❖ 理大専用のレポート用紙に書くこと。

❖ ホッチキスまたは糊で綴じること。

❖ 学生番号、氏名、講義名、提出日を書くこと。

❖ 途中の計算過程を書くこと。

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

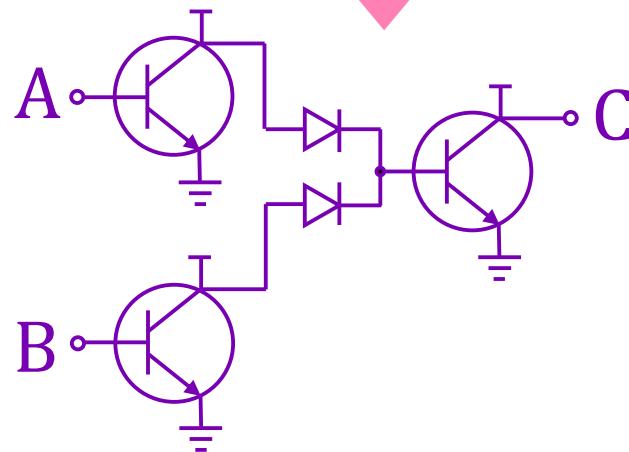
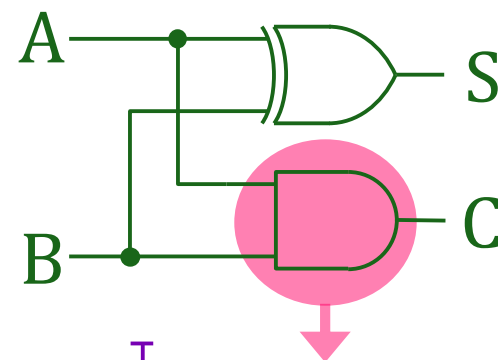
演算回路の設計

① 回路化したい計算式 $d = a + b$

② 論理式 $S = A \oplus B$
 $C = A \cdot B$

③ 論理回路

④ デジタル回路



論理演算(ブール代数)

真 (true) と偽 (false) の2状態を扱う
演算

重要!

論理値

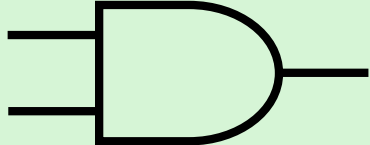
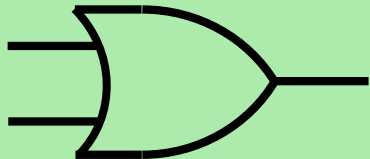
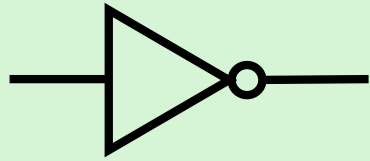
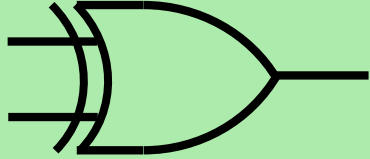
真 ... 1

偽 ... 0

2進数 (0, 1) の演算の実現に適している。

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	$A + B$	
否定 NOT	\bar{A}	
排他的論理和 Exclusive OR (XOR)	$A \oplus B$	

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

- ① 真理値表の作成
- ② 論理式の組み立て
- ③ 論理式の簡単化
- ④ 論理回路への置き換え

半加算器

真理值表

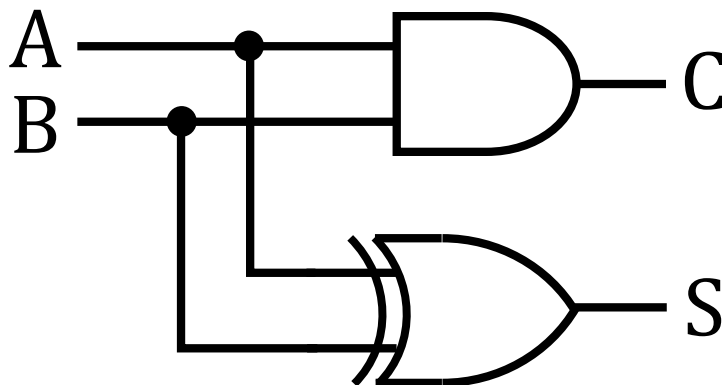
A	B	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

論理式

$$C = A \cdot B$$

$$S = A \oplus B$$

論理回路



全加算器

真理值表

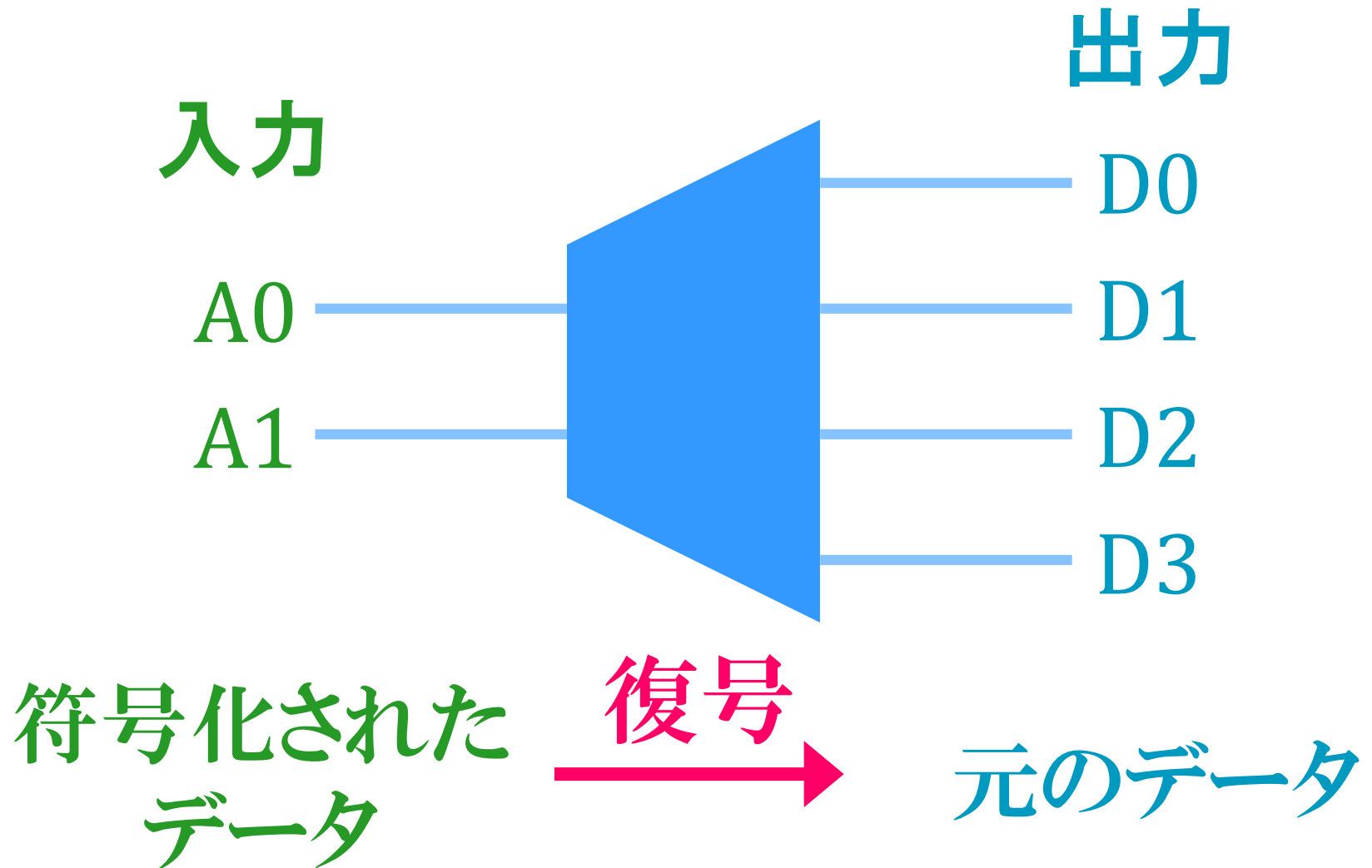
A	B	Z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

論理式

$$\begin{aligned} C &= \bar{A} \cdot B \cdot Z + A \cdot \bar{B} \cdot Z + \\ & A \cdot B \cdot \bar{Z} + A \cdot B \cdot Z \\ &= A \cdot B + B \cdot Z + A \cdot Z \end{aligned}$$

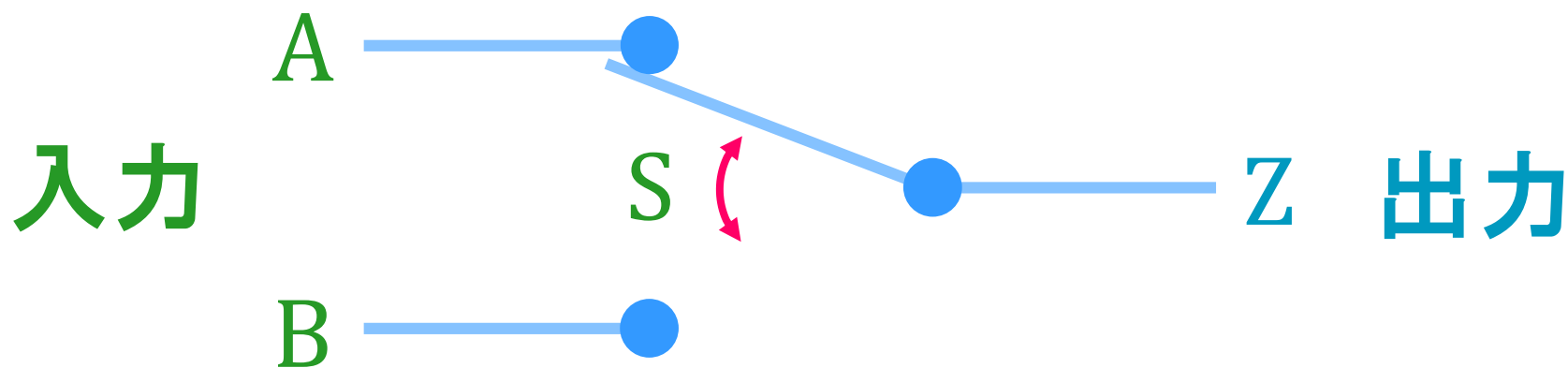
$$\begin{aligned} S &= \bar{A} \cdot \bar{B} \cdot Z + \bar{A} \cdot B \cdot \bar{Z} + \\ & A \cdot \bar{B} \cdot \bar{Z} + A \cdot B \cdot Z \end{aligned}$$

2入力4出力デコーダ



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

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

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

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

$$D3 = A0 \cdot A1 \quad Z = \overline{S} \cdot A + S \cdot B$$

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

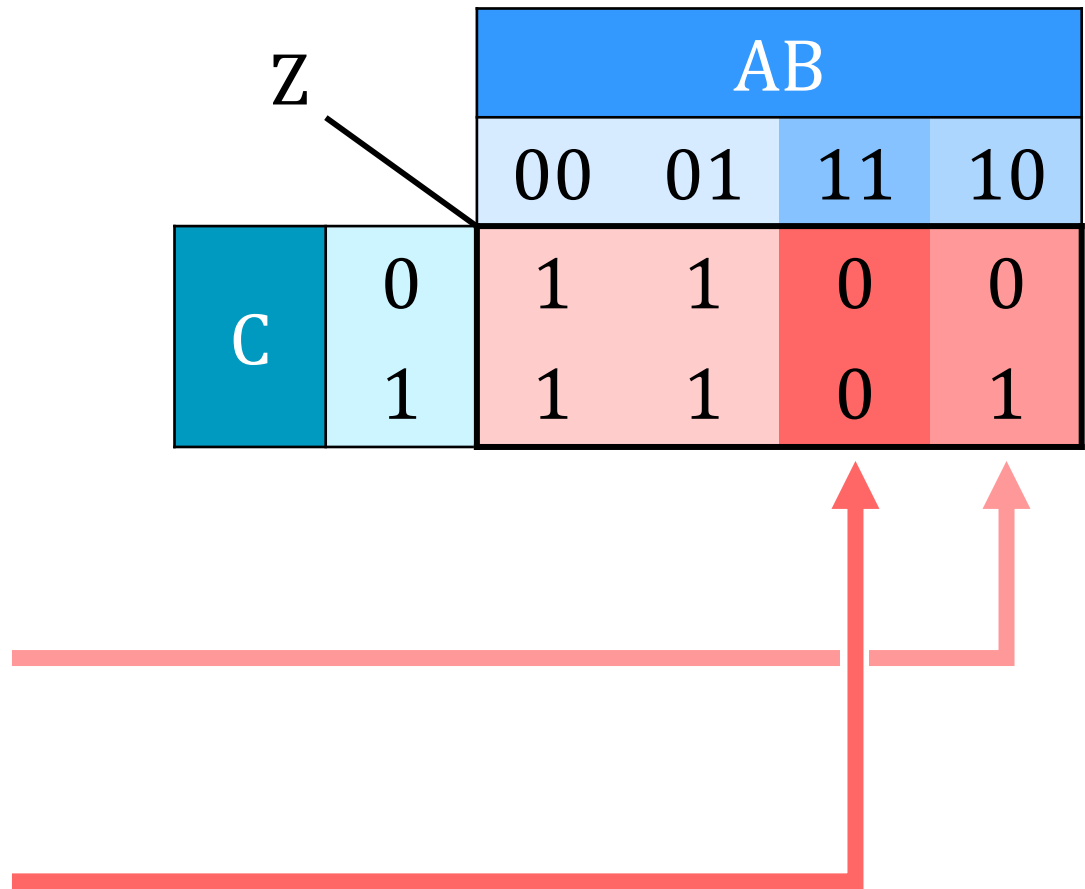
カルノー図

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

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。

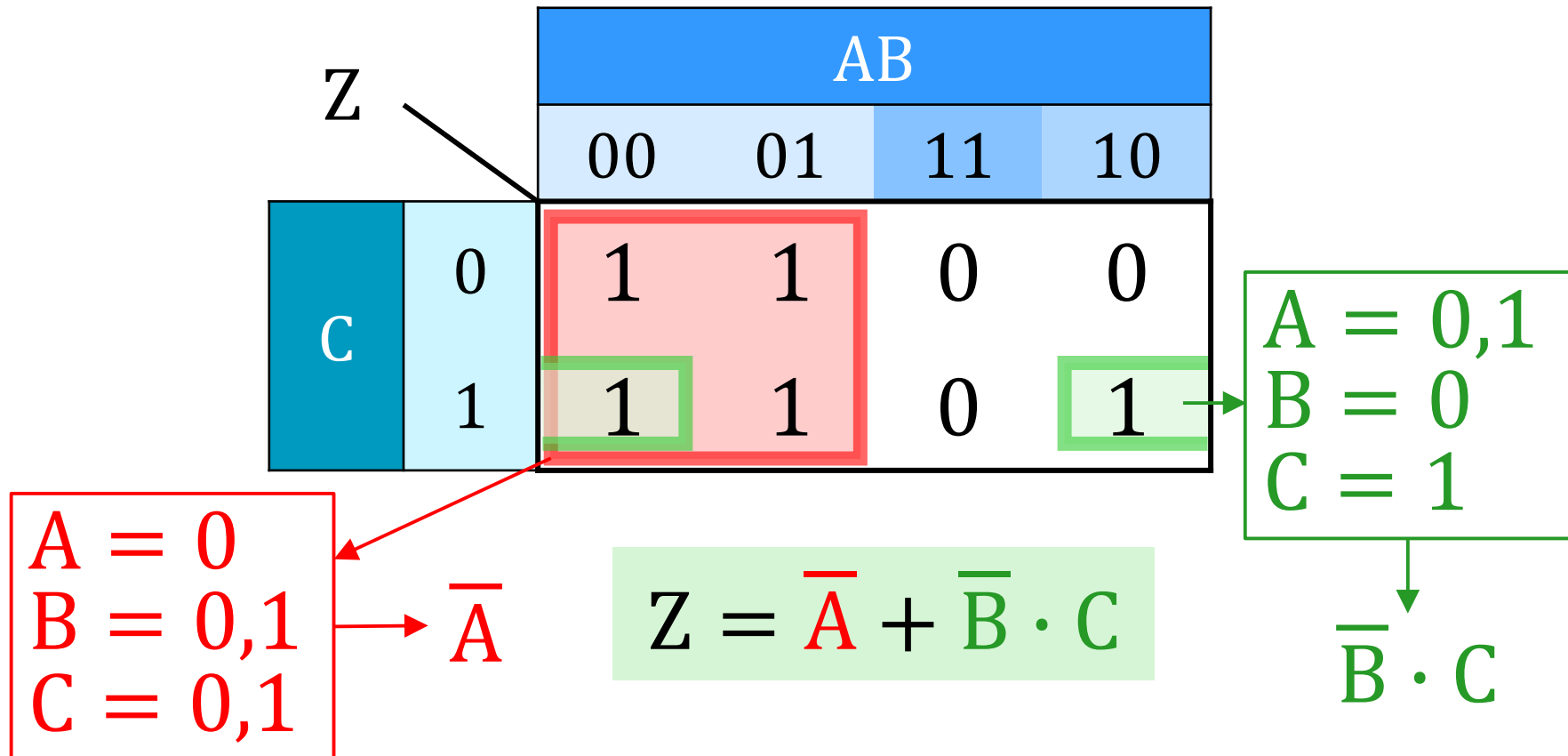
Z		AB			
		00	01	11	10
C	0	1	1	0	0
	1	1	1	0	1

上下、左右は繋がっていると考え、四角形が重なっても良い。

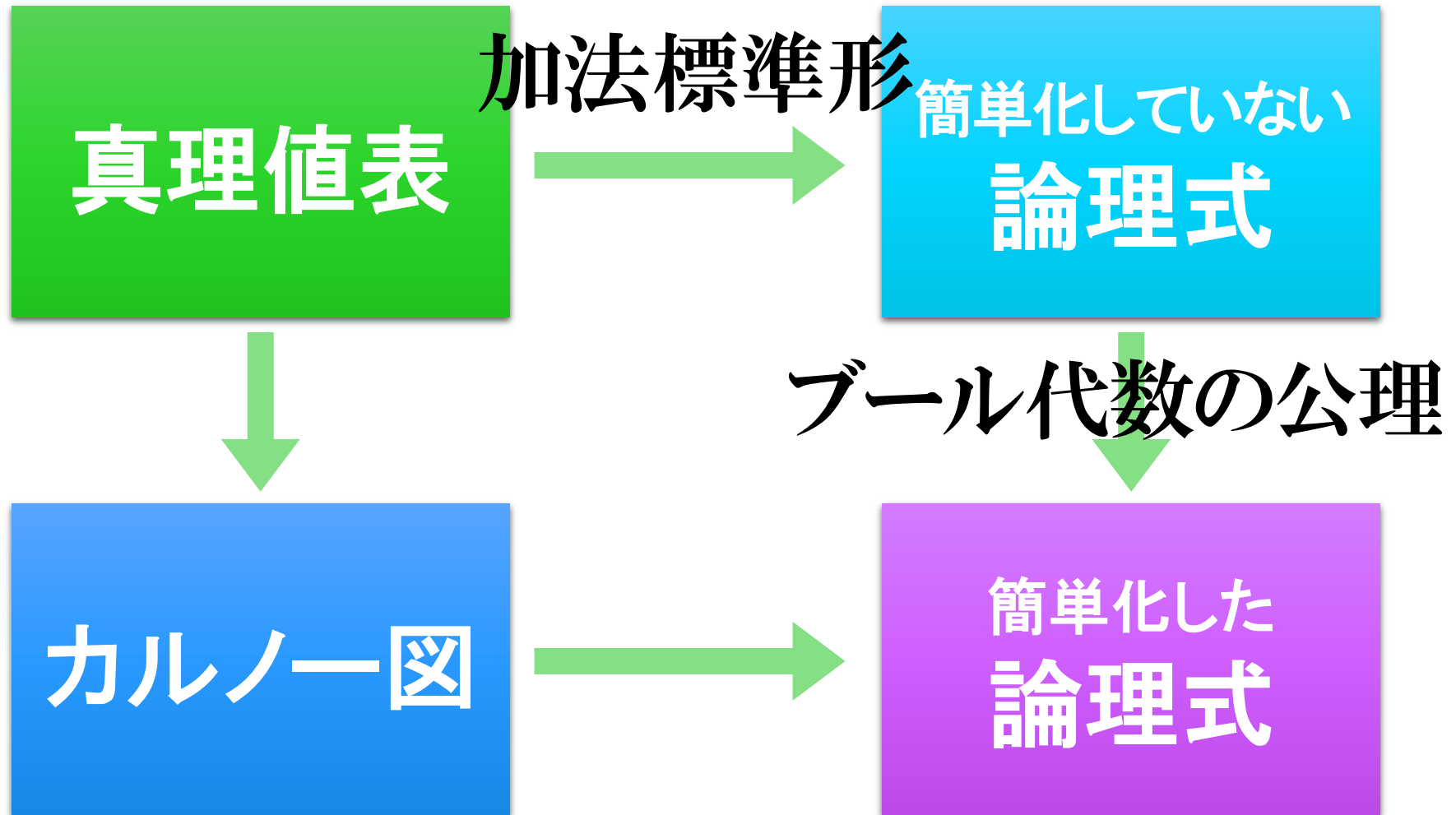
カルノー図

重要!

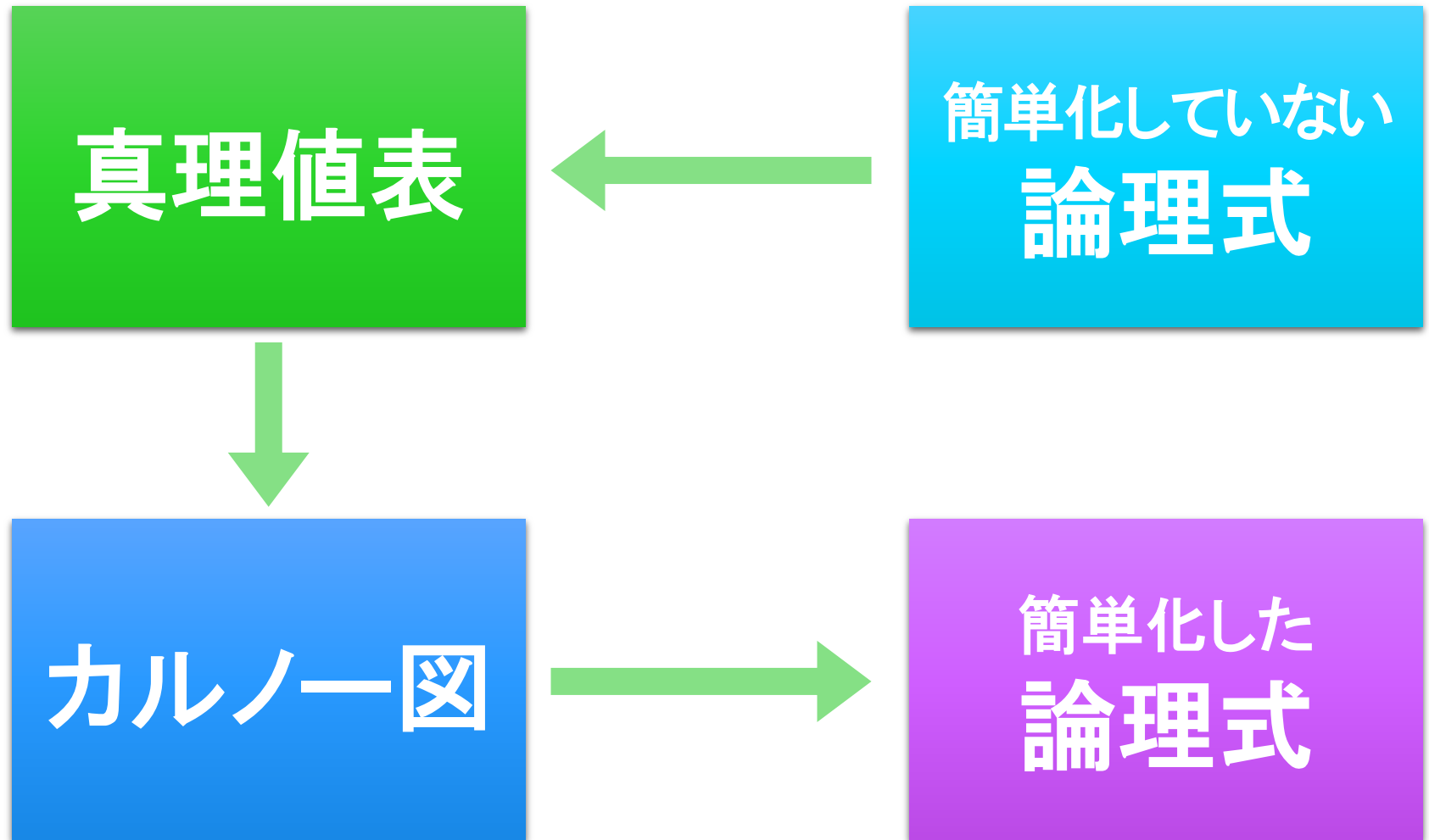
- ① 各四角形において、入力値が 0 または 1 の片方しか含まれていない変数で、論理積項を表す。
- ② すべての論理積項の論理和を求める。



カルノー図による論理式の簡単化



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: *月*日(*) 講義開始前

- ❖ 理大専用のレポート用紙に書くこと。
- ❖ ホッチキスまたは糊で綴じること。
- ❖ 学生番号、氏名、講義名、提出日を書くこと。
- ❖ 途中の計算過程を書くこと。

中間試験 12月21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!

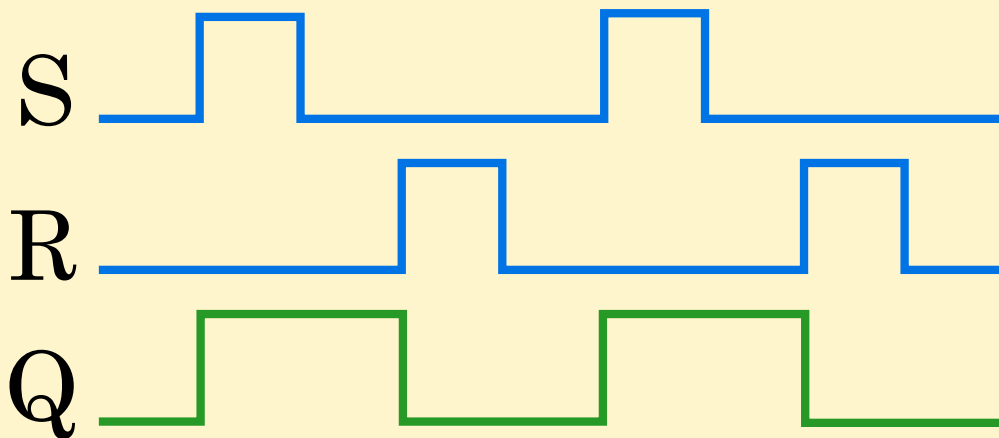
入力信号

R リセット S セット

出力信号

$Q_{(n)}$ 時刻 n のときの出力

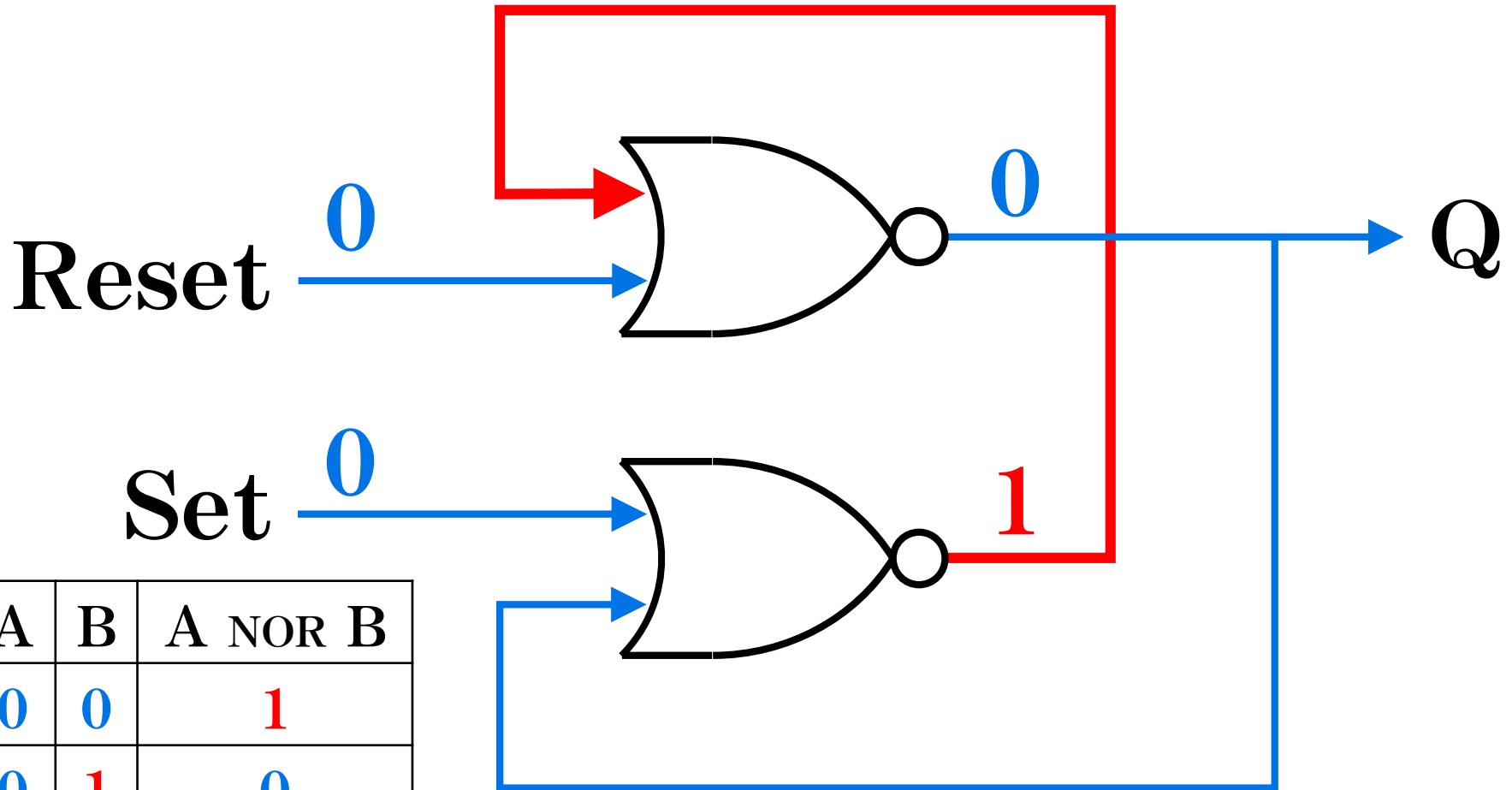
タイミングチャート



動作表

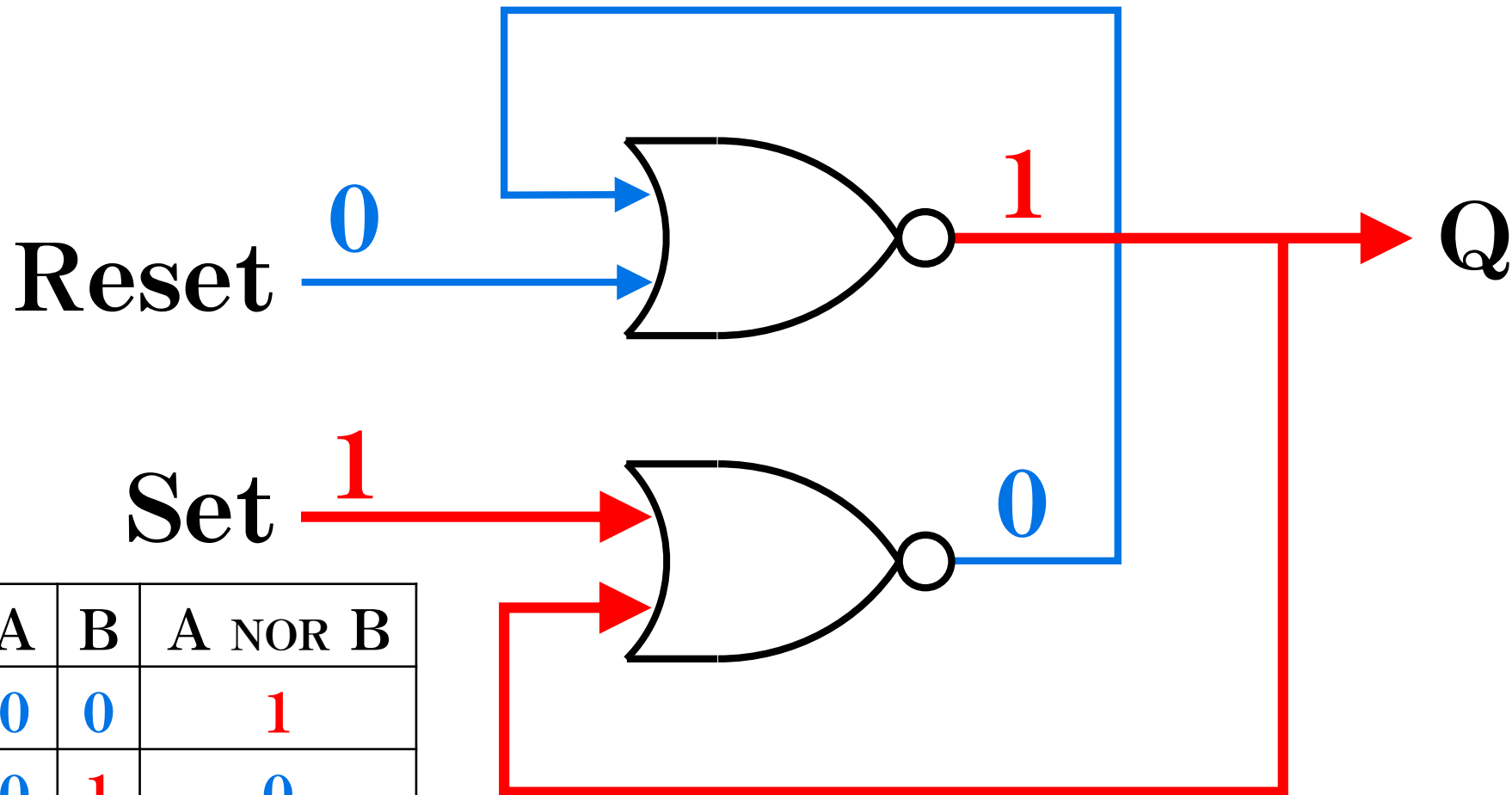
R	S	$Q_{(n)}$
0	0	$Q_{(n-1)}$
0	1	1
1	0	0
1	1	不定

フリップフロップの仕組み



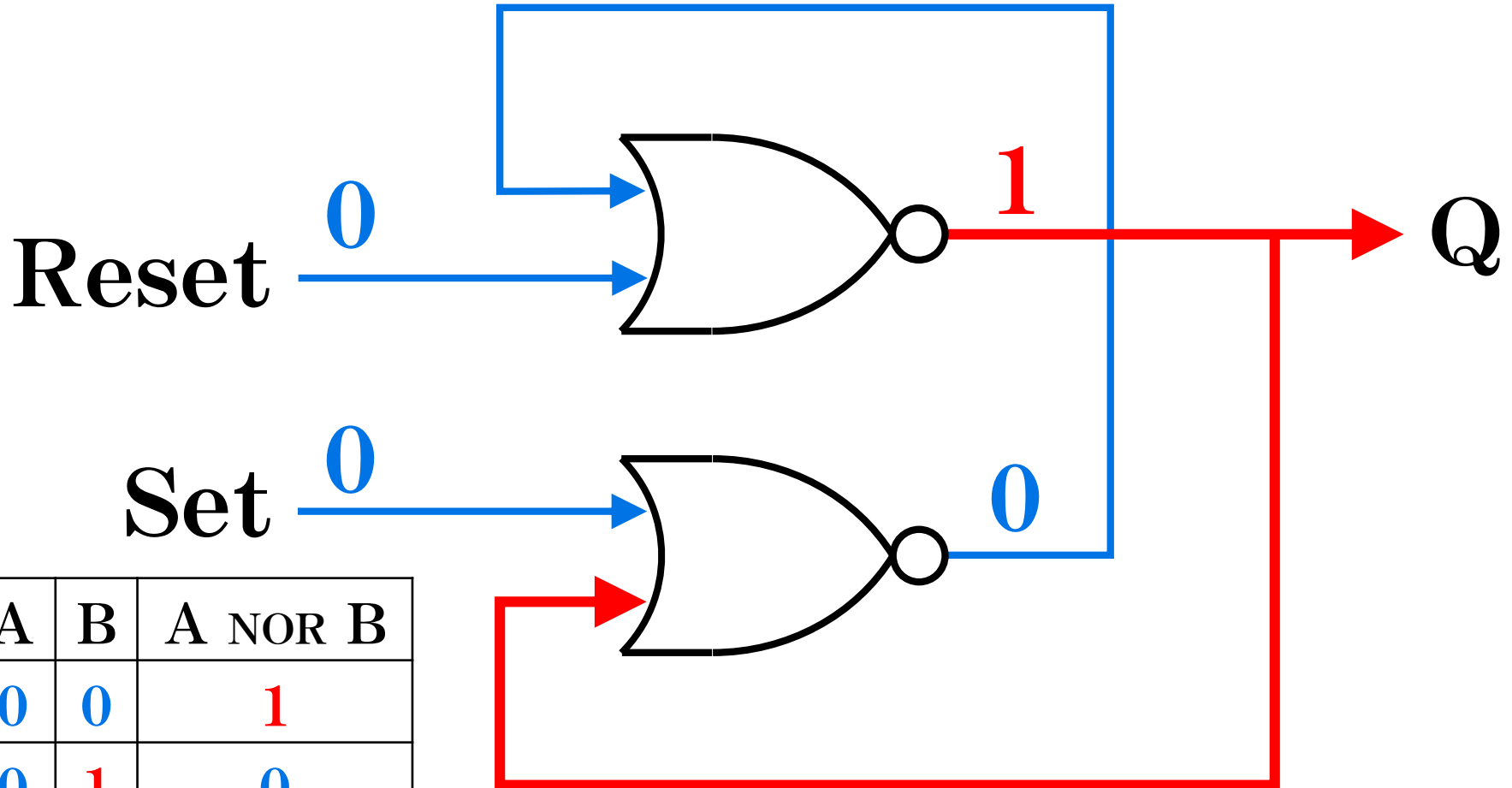
A	B	A NOR B
0	0	1
0	1	0
1	0	0
1	1	0

フリップフロップの仕組み



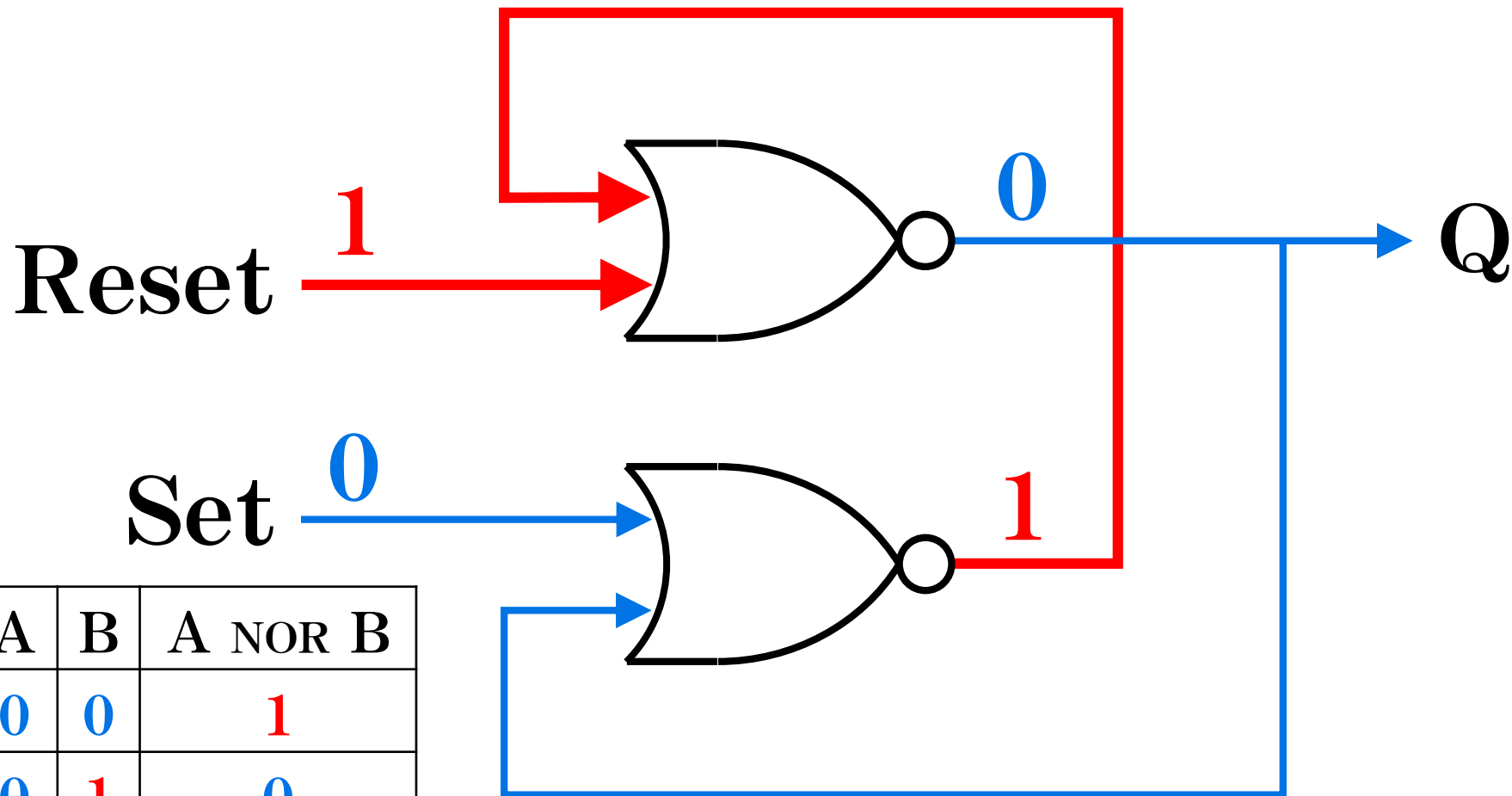
A	B	A NOR B
0	0	1
0	1	0
1	0	0
1	1	0

フリップフロップの仕組み



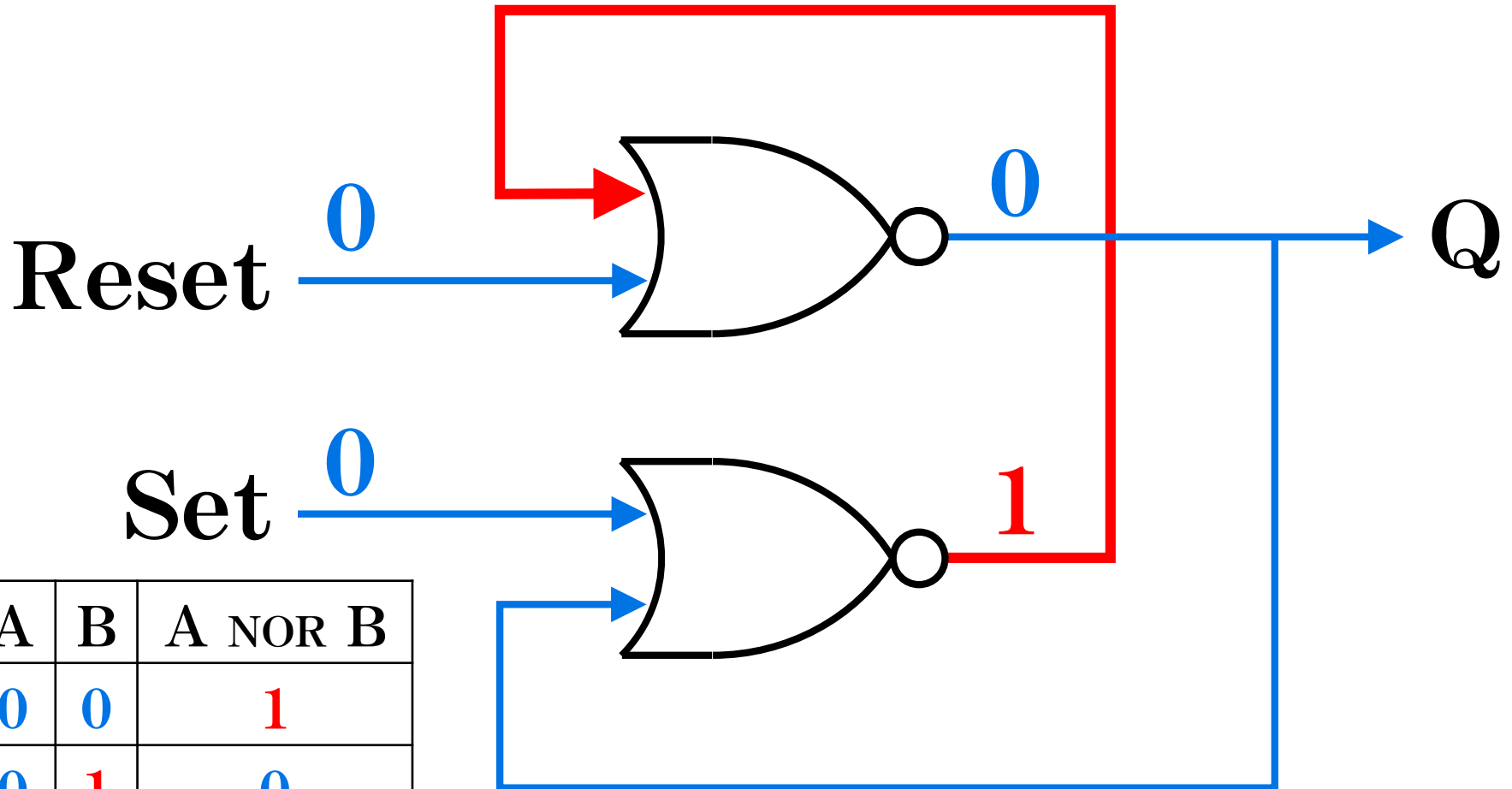
A	B	A NOR B
0	0	1
0	1	0
1	0	0
1	1	0

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

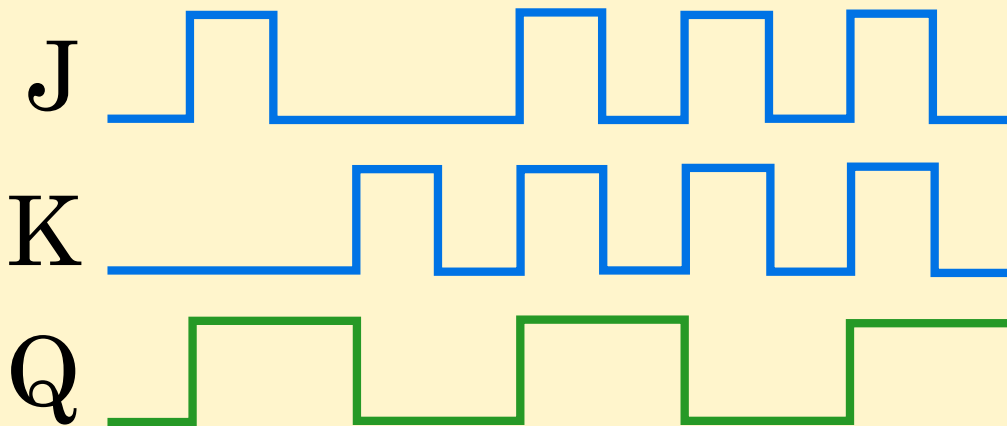
JKフリップフロップ

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

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

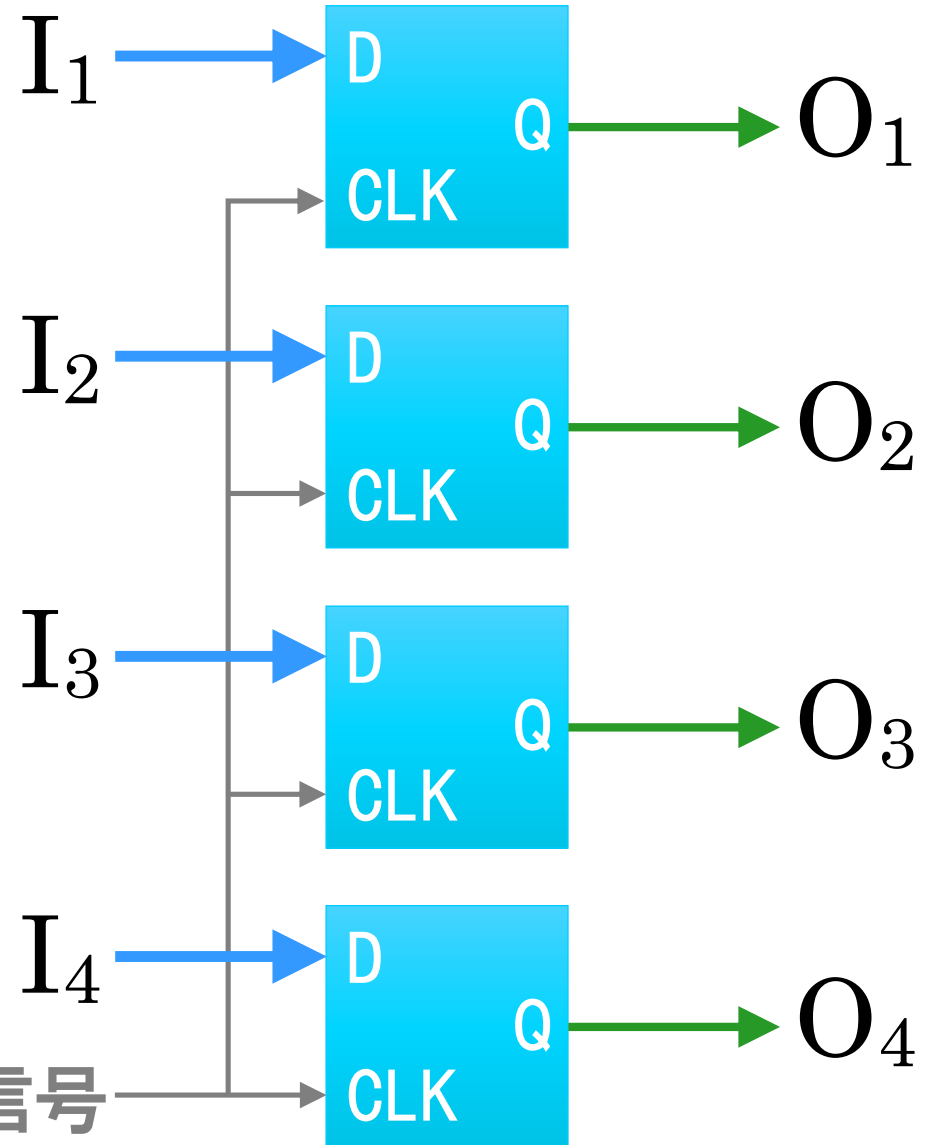
✦ 基本記憶素子(キャッシュメモリ)

レジスタ

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

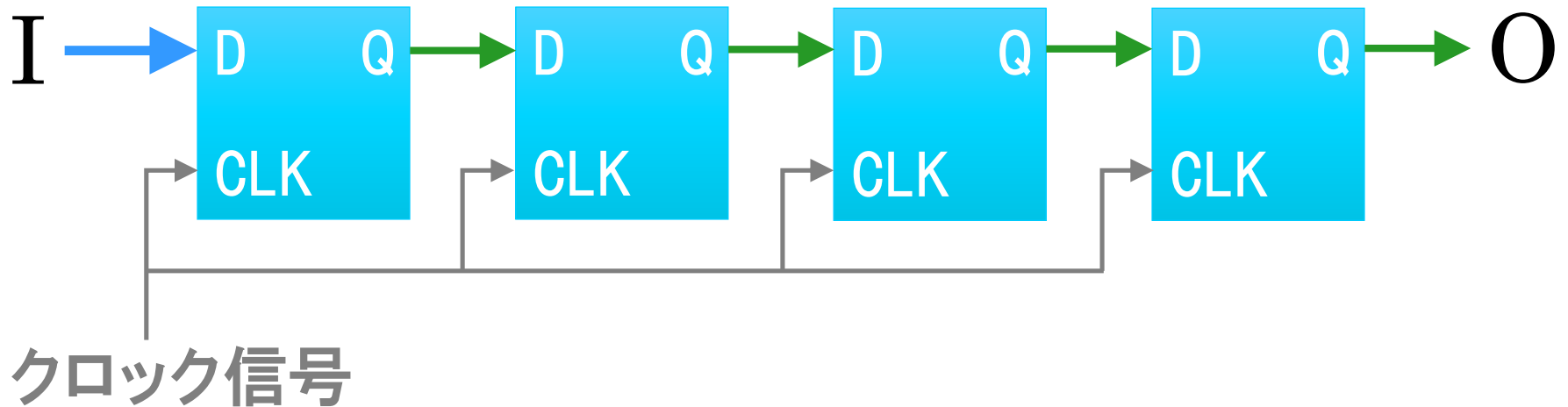
クロック信号



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

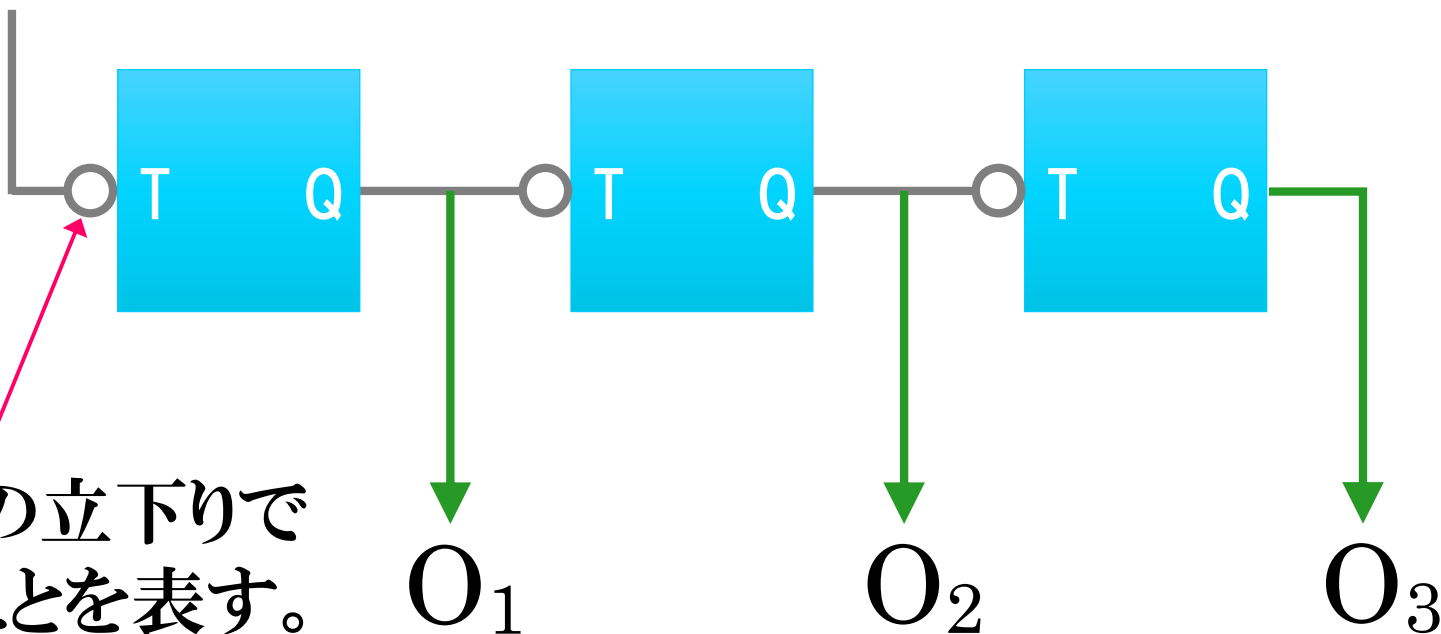
データを1bitずつ送信する(シリアル通信)。



カウンタ

入力信号のパルス数をかぞえる機能を持つレジスタ

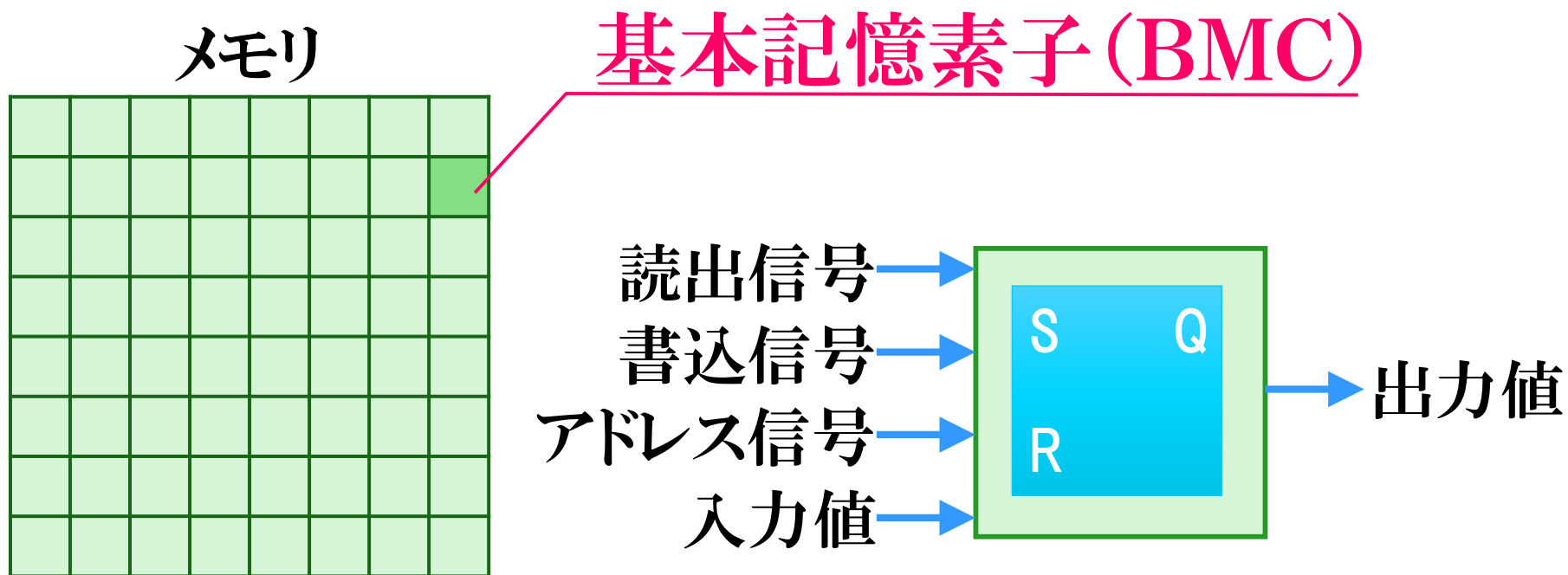
クロック信号



入力信号の立下りで動作することを表す。

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。