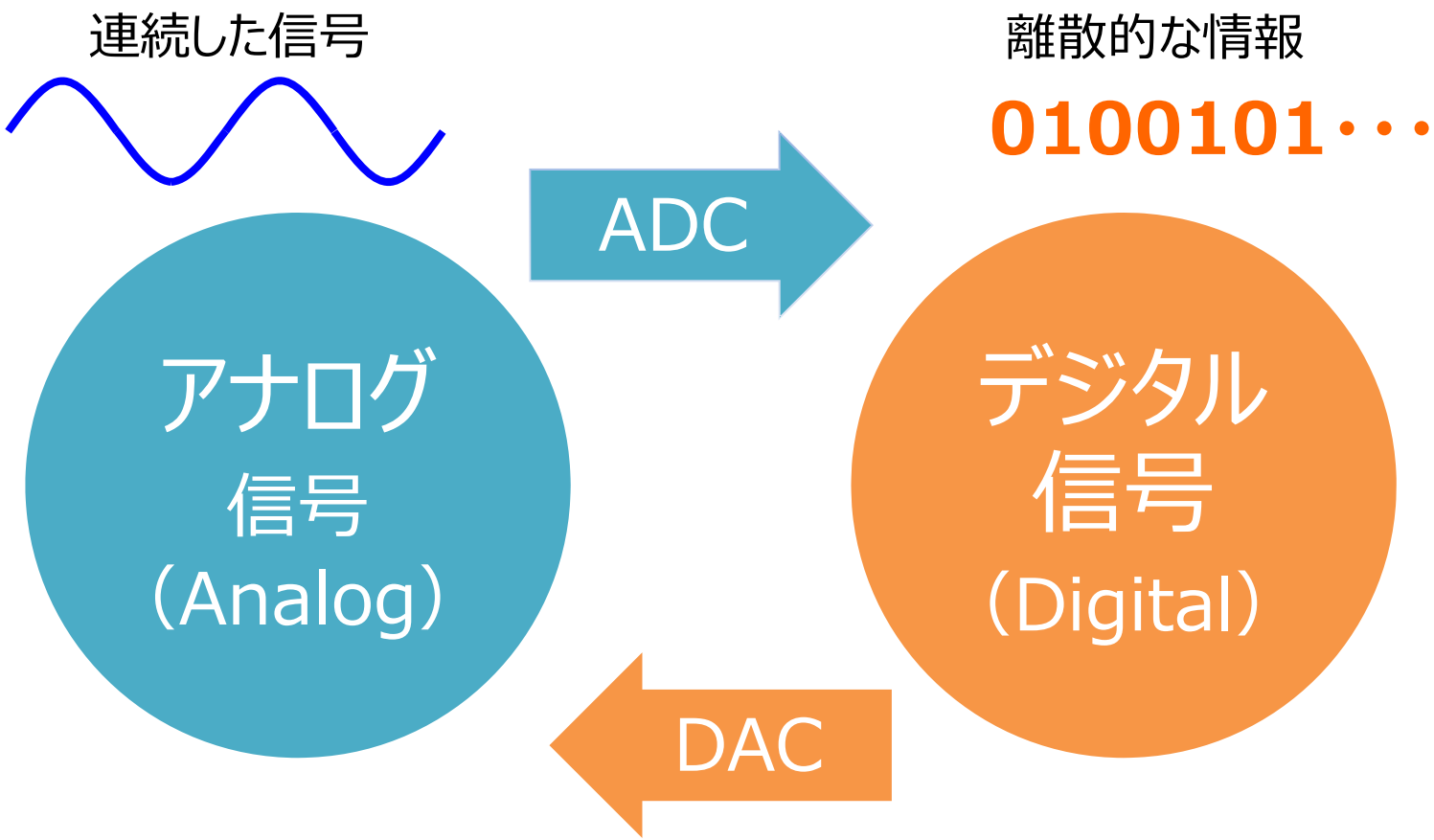
アナログ信号とデジタル信号の関係

NO.1



【ADCの手順】 サンプリング ⇒ 量子化 ⇒ 符号化

デジタル回路は回路上で2つの不連続な電位範囲を情報の表現に用いる電子回路

⇒デジタル信号(0もしくは1）を扱う回路

デジタル回路に必要な基礎知識

ブール代数とは？

ブール代数（ブールだいすう、Boolean algebra）とは論理変数と論理関数（演算）を扱う代数。AND、OR、NOTを以下のような記号で表現する。

論理	記号	記述例	記述例の意味
OR	＋	A＋B	AもしくはB
AND	・	A・B	AかつB
NOT	―	$\overline{A}$	Aの反転

【ドモルガンの法則】

論理積(AND)、論理和(OR)、否定(NOT)の間に成り立つ関係を表す定理。真・偽を反転させ、論理和・論理積の定義を交換した論理が元の論理と同じになることを示している。

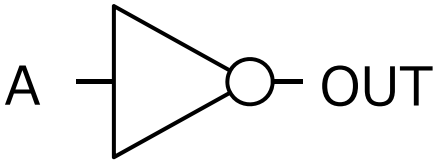
（ドモルガンの法則の例）

$\bar{A}\cdot\bar{B} = \overline{A+B}$ 、 $\overline{A\cdot B} = \bar{A}+\bar{B}$、

代表的な論理回路1

【否定論理（NOT）】

＜シンボル＞



$OUT = \overline{A}$

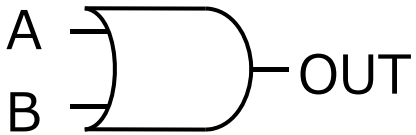
＜真理値表＞

A	OUT
0	1
1	0

真理値表（しんりちひょう、Truth table）は論理式および論理回路において、すべての入出力の関係を表にしたもの。

【論理和（OR）】

＜シンボル＞



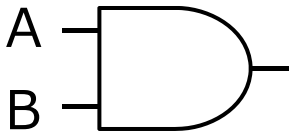
$OUT = A + B$

＜真理値表＞

A	B	OUT
0	0	0
0	1	1
1	0	1
1	1	1

【論理積（AND）】

＜シンボル＞



$OUT = A \cdot B$

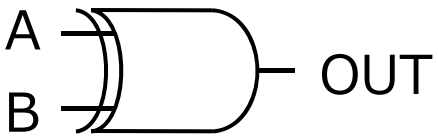
＜真理値表＞

A	B	OUT
0	0	0
0	1	0
1	0	0
1	1	1

代表的な論理回路2

【排他的論理和（XOR）】

＜シンボル＞



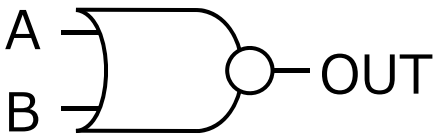
$OUT = A \oplus B$

＜真理値表＞

A	B	OUT
0	0	0
0	1	1
1	0	1
1	1	0

【論理和の否定（NOR）】

＜シンボル＞



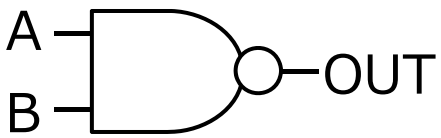
$OUT = \overline{A + B}$

＜真理値表＞

A	B	OUT
0	0	1
0	1	0
1	0	0
1	1	0

【論理積の否定（NAND）】

＜シンボル＞



$OUT = \overline{A \cdot B}$

＜真理値表＞

A	B	OUT
0	0	1
0	1	1
1	0	1
1	1	0

カルノー図

カルノー図（カルノーず、Karnaugh map）は論理式を簡単化するための表。論理式を簡略化することで回路に使う素子数を減らすことができる。

【カルノー図を使ったデジタル回路の簡略化手順】

①真理値表のように、カルノー図に「１」を書き込む

②隣接する1をループでくくる。できるだけループの数が少なくなるようにくくる

（※ループは縦・横の長方形でくくる。１・２・４個単位でくくらないといけない。）

③くくったループに対応論理式を書きだす

①カルノー図のフォーマット

【3入力（ＡＢＣ）に対応したカルノーズ】

C \ AB	00	01	11	10
0				
1				

→ 00→01→11→10　と両隣が1bit違いになるように並べる
00→11　等の並びはダメ

【4入力（ＡＢＣＤ）に対応したカルノーズ】

CD \ AB	00	01	11	10
00				
01				
11				
10				

→ 00→01→11→10　と両隣が1bit違いになるように並べる
00→11　等の並びはダメ

②ループでくくる時のルール

【ダメな例】

1,2,4個単位でくくらないといけない

CD \ AB	00	01	11	10
00				
01	1	1	1	
11				
10				

長方形以外はダメ

CD \ AB	00	01	11	10
00		1		
01	1	1		
11		1		
10				

斜めでくくるのはダメ

CD \ AB	00	01	11	10
00				
01		1		
11			1	
10				

【良い例】

2度同じ箇所をくくるのは問題無し

CD \ AB	00	01	11	10
00				
01	1	1	1	
11				
10				

2個単位でくくる

CD \ AB	00	01	11	10
00		1		
01	1	1		
11		1		
10				

1個単位でくくる

CD \ AB	00	01	11	10
00				
01		1		
11			1	
10				

カルノー図の使用例1

【例題】 以下のブール代数を簡単にせよ。

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

【手順①】 真理値表のように、カルノー図に「1」を書き込む

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

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

例えば $A \cdot B \cdot \overline{C}$ は、
A=1、B=1、C=0 に該当する箇所に1を記入

【手順②】 隣接する1をループでくくる。できるだけループの数が少なくなるようにくくる

（※ループは縦・横の長方形でくくる。1・2・4個単位でくくらないといけない。）

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

【手順③】 くくったループに対応論理式を書きだす

$f = A$

カルノー図の使用例2

【例題】 以下のブールを簡単にせよ。

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

【手順①】 真理値表のように、カルノー図に「1」を書き込む

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

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

【手順②】 隣接する1をループでくくる。できるだけループの数が少なくなるようにくくる

（※ループは縦・横の長方形でくくる。1・2・4個単位でくくらないといけない。）

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

【手順③】 くくったループに対応論理式を書きだす

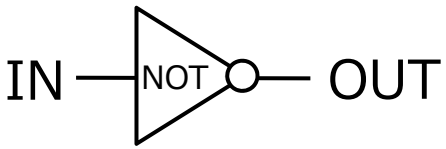
$$f = B + \overline{C}$$

基本的なデジタル回路
（フリップフロップ）

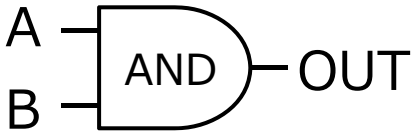
組み合わせ回路と順序回路

【組み合わせ回路（）】

現在の入力だけ（入力信号の組み合わせだけ）で出力が決まるデジタル回路
例．NOT、AND、OR、NAND、NOR　等々



$OUT = \overline{IN}$

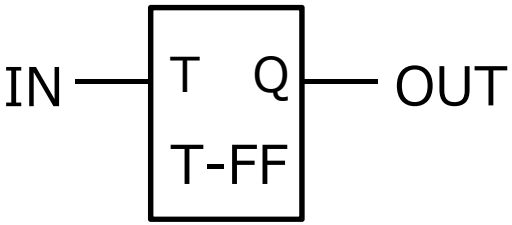


$OUT = A \cdot B$

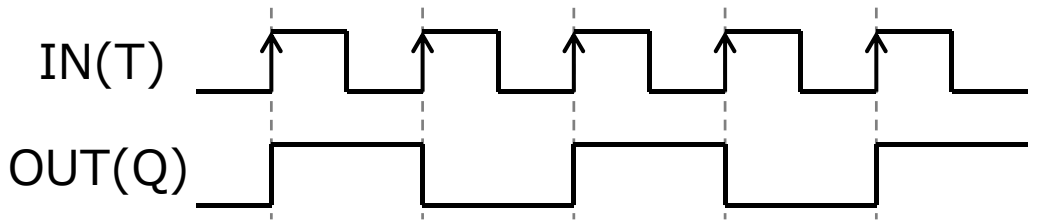
出力が過去の状態に
影響されない

【順序回路（）】

現在の入力と過去の入力で出力が決まるデジタル回路　→メモリとして利用可能
例．T-FF、JK-FF、D-FF　等々



<T-FF のタイミングチャート>



入力の立ち上がり毎に出力が反転(←前の状態に依存)

フリップフロップとは？

フリップフロップ（Flip-Flop）は論理回路で構成され、記憶機能を備えている。例えばコンピュータの記憶装置を構成する回路（SRAM）として使用される。

論理回路ではなく、キャパシタの充電状態を使用した記憶方式（DRAM）と比較し、コンデンサの自然放電（リーク電流）によるデータ消失を防ぐための定期的なリフレッシュ動作（再書き込み）を必要とせず、記憶領域へ高速なアクセスが可能。ただし、1bitあたりのトランジスタ数やその配線がDRAMと比べ複雑になるため、コストは高くなる。

フリップフロップの利用と分類

【SRAMとDRAMの使い分け】

比較的小容量で高速性が求められる（CPUのレジスタ等）の揮発性RAM領域においては主にフリップフロップベースのSRAMを採用し、比較的低速で大容量が求められる一般的な揮発性RAM領域において主にDRAMが利用される。

それぞれの特長を活かした使い分けが行われている。

【フリップフロップの分類】

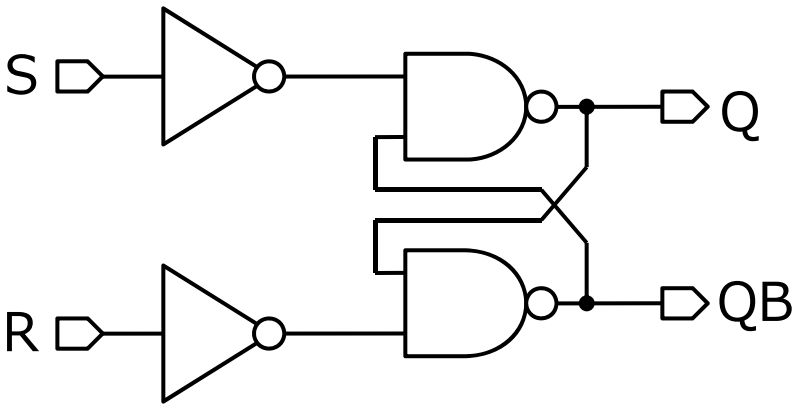
SR型（RS型とも呼ばれる。Set-Resetの略）

JK型（語源不明）、D型（Delayedの略）、T型（Toggleの略）

非同期－SR型フリップフロップ

SRという名称は、セット (Set) と リセット (Reset) の2入力の信号名に由来する。SR型は同時にRとSに入力があると、結果が不定になる（出力が正しく定まらない）。

【回路図】



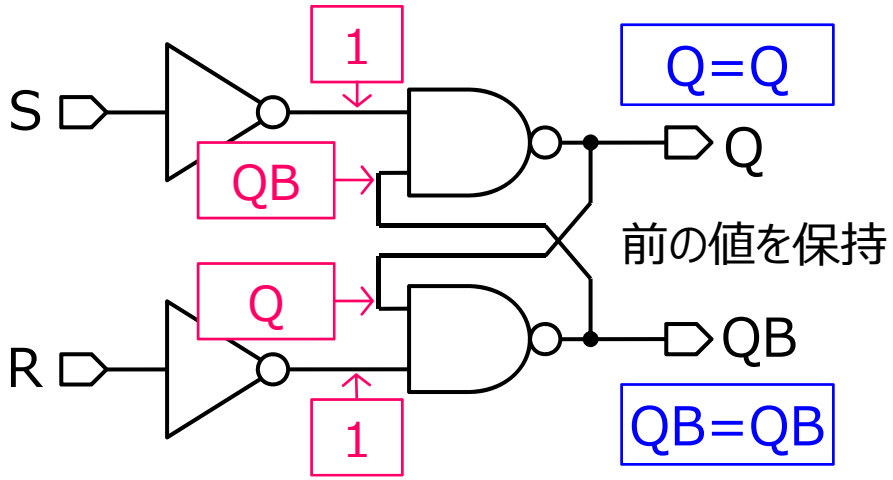
【真理値表】

S	R	Q	QB
0	0	過去の値を保持	過去の値を保持
0	1	0	1
1	0	1	0
1	1	禁止	禁止

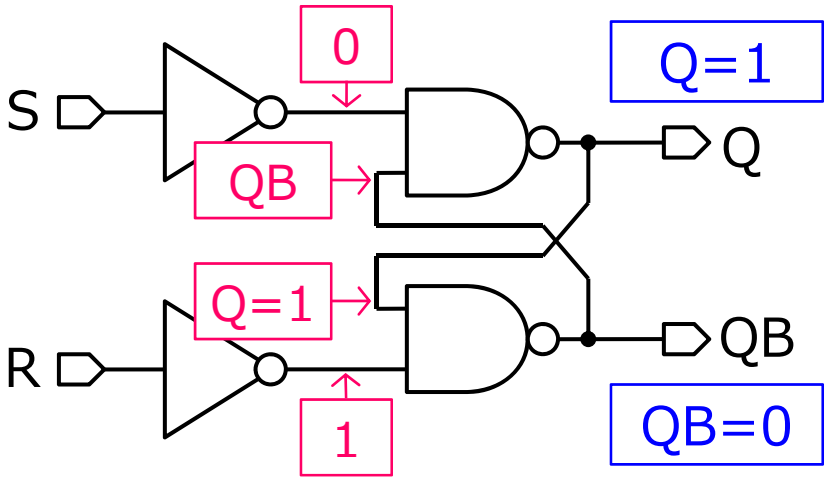
※QBはQの反転

非同期－SR型フリップフロップの動作

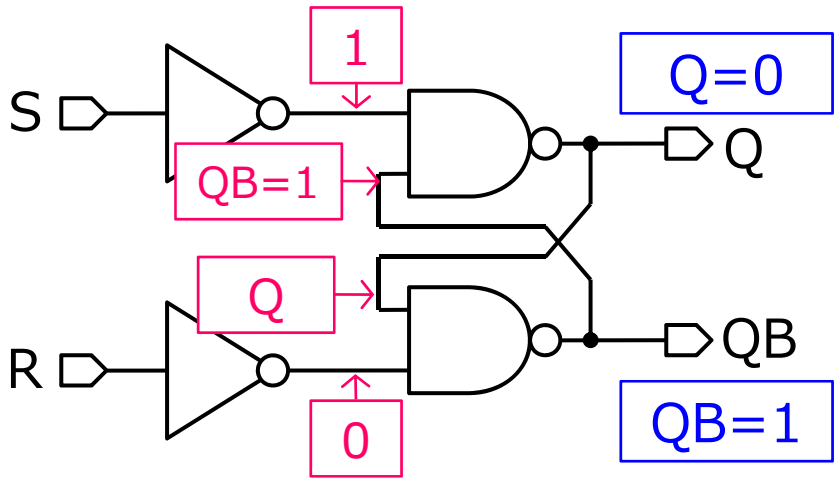
【S=0、R=0】



【S=1、R=0】

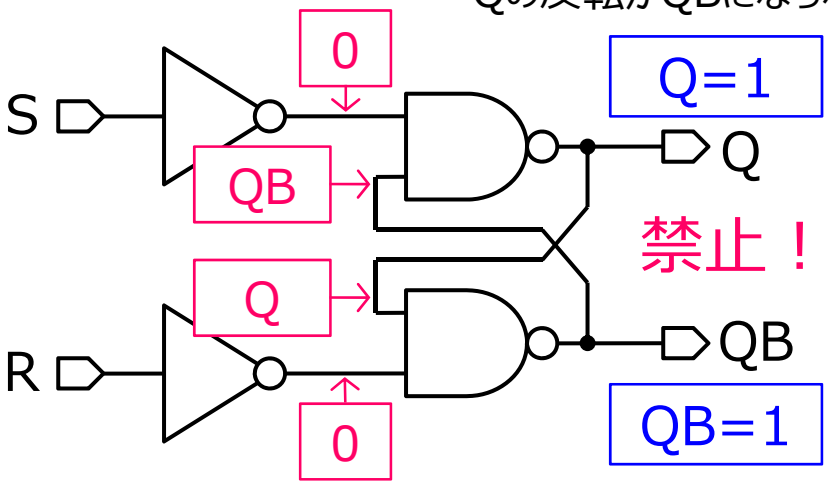


【S=0、R=1】



【S=1、R=1】

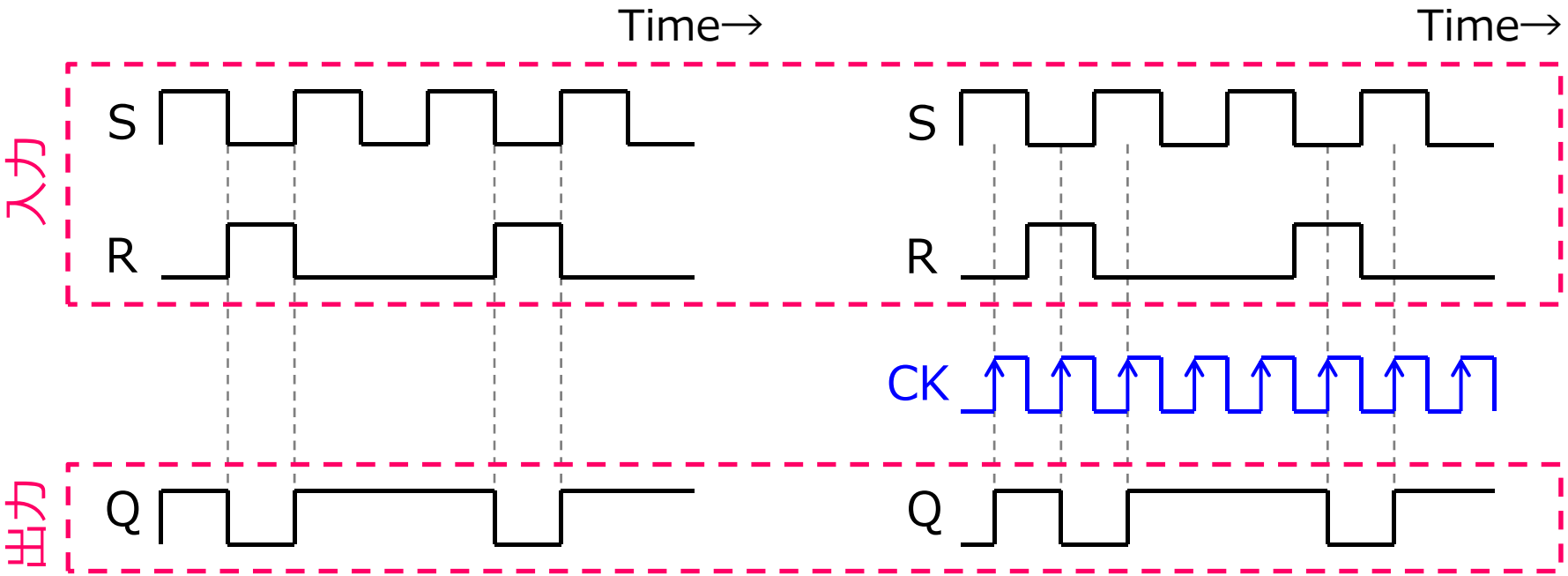
Qの反転がQBにならない



非同期回路と同期回路

【非同期-SR型FF】

【同期-SR型FF】



入力の変化に応じて即座に出力が変化する

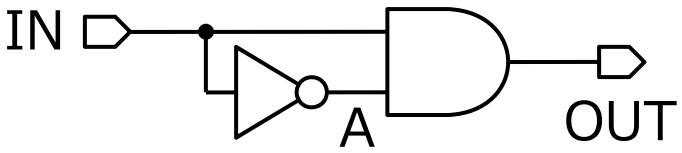
入力の変化に応じてクロック(CK)の立ち上がり(もしくは立下り)で出力が変化する

デジタル回路の多くは同期回路で設計される
⇒理由はハザードが発生しない、出荷時テストが容易（CADが整っている）なため

ハザード

ハザード・・・論理上は発生しないが、回路遅延により発生するパルス

【回路例】

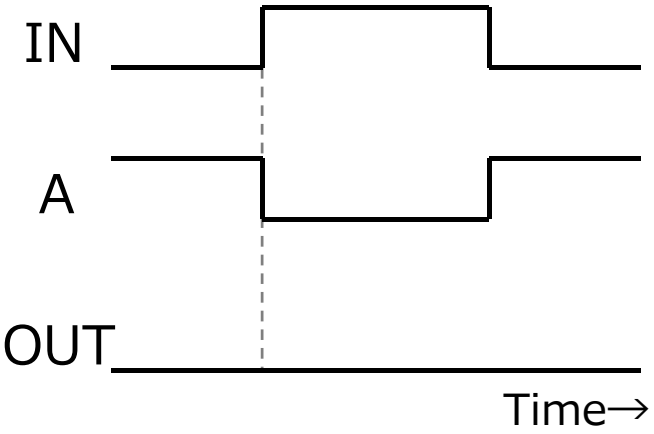


・非同期回路はハザードが発生するので注意

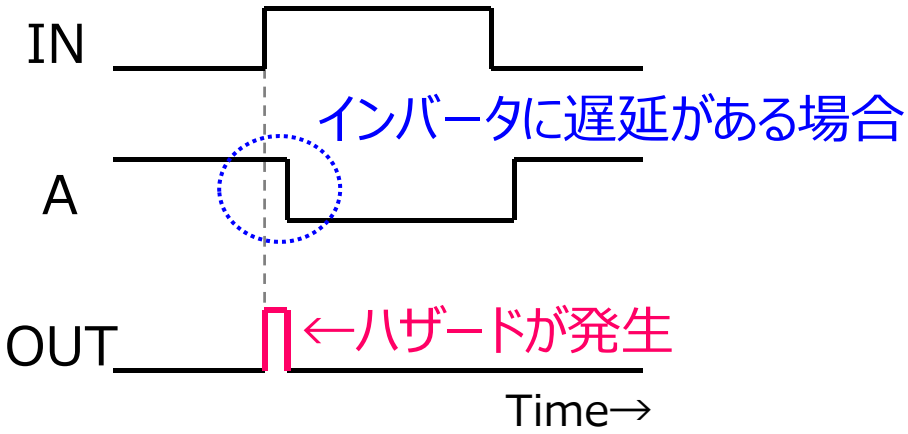
・同期回路はハザードが問題にならない

（＝問題にならないよう設計可能）

【遅延なし(理想)】



【遅延あり(現実)】



JK型フリップフロップ（JK-FF）

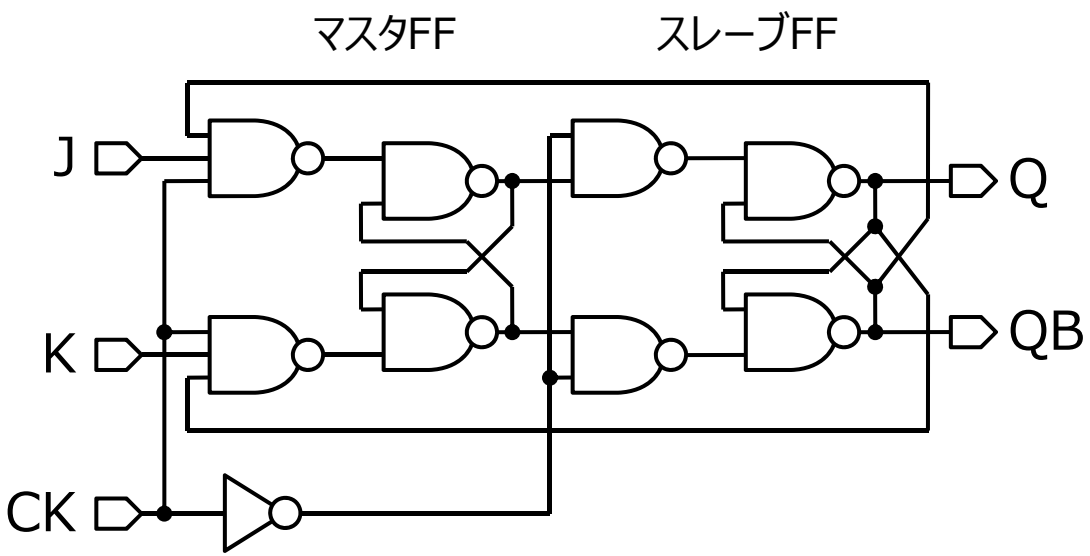
SR型フリップフロップと似た動作をする。

SR型と同じ点・・・J（=S）でセット、K（=R）でリセットされる。

SR型と違う点・・・CKの立ち上がり（もしくは立下り）で出力が変化

２入力が共にHの場合はクロック毎に出力が反転する。

【回路図（例）】

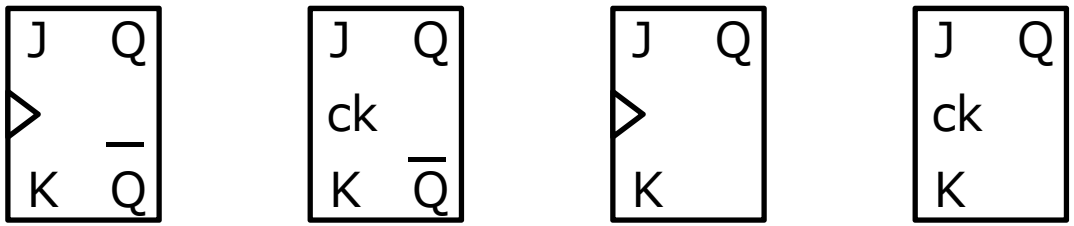


【真理値表】

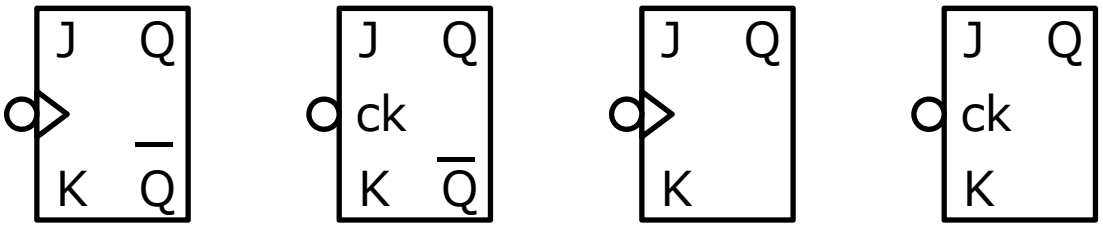
J	K	Q	$\overline{Q}$
0	0	保持	保持
0	1	0	1
1	0	1	0
1	1	反転	反転

JK型フリップフロップのシンボル（記号）

【CLKの立ち上がりで出力が変化するもの】



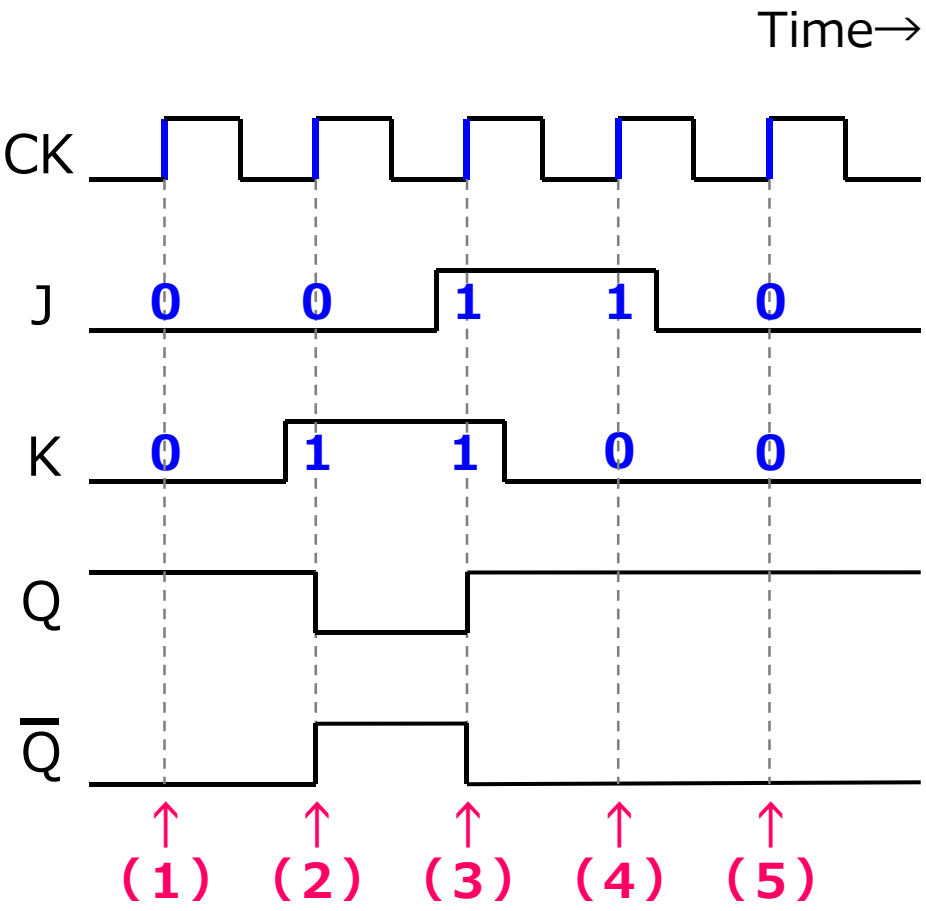
【CLKの立ち下がりで出力が変化するもの】



※デジタル回路のシンボルにおいて○がつくと反転を意味する。

JK型フリップフロップの動作

CK（クロック）の立ち上がりで出力が変化するJK型フリップフロップ



（1）J=0、K=0だから、
Qには“1”が保持

（2）J=0、K=1だから、
Qは“0”にリセット

（3）J=1、K=1だから、
Qは“1”に反転

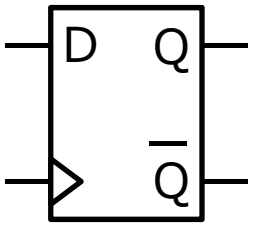
（4）J=1、K=0だから、
Qには“1”がセット

（5）J=0、K=0だから、
Qには“1”が保持

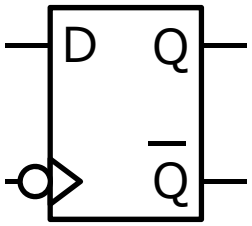
D型フリップフロップ（D-FF）

CKの立ち上がり(もしくは立下り)時に入力端子Dの値を読み、その値を出力Qに設定する。

【シンボル】

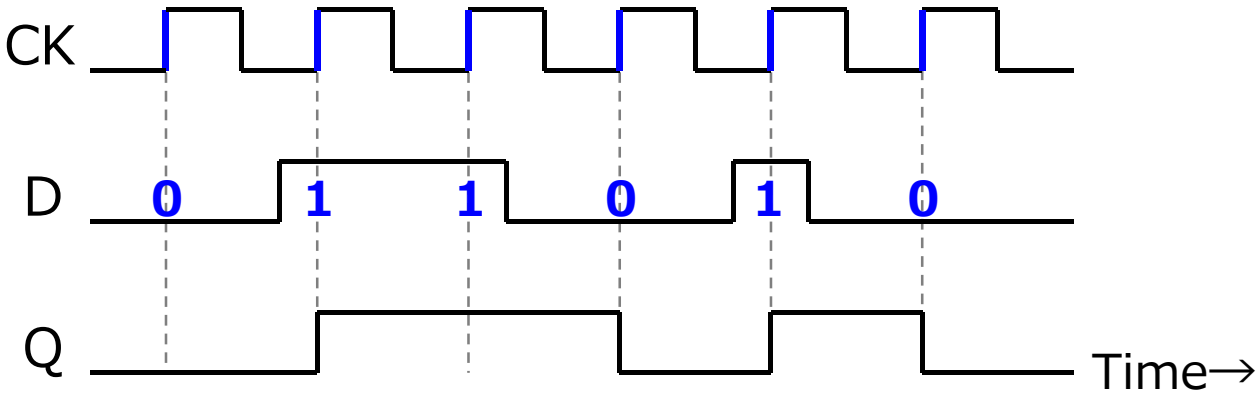


（a）立ち上がりエッジ型



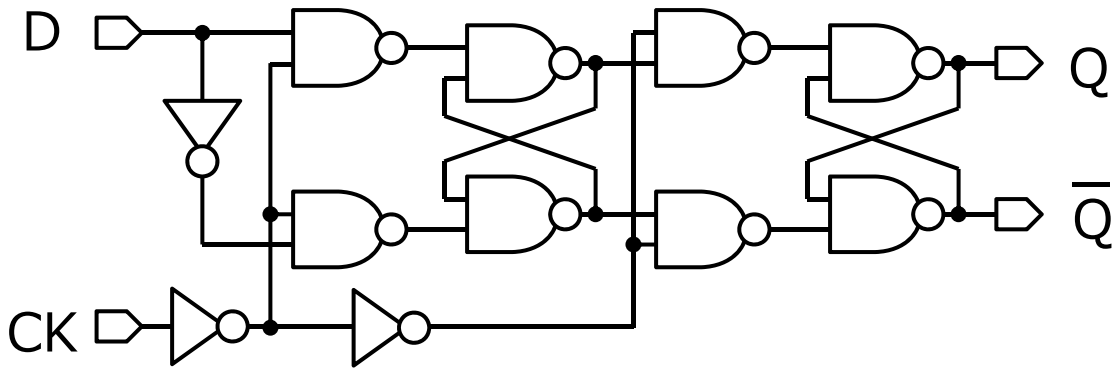
（b）立ち下がりエッジ型

【動作（CKの立ち上がりで変化するタイプ）】



D型フリップフロップの回路例

【回路構成例１】

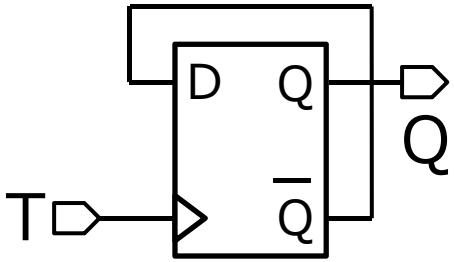


【回路構成例２】

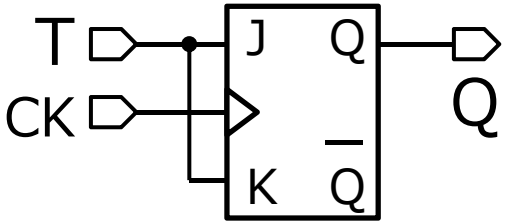
T型フリップフロップ（T-FF）

1つの入力T（Trigger）を持ち、入力Tの立ち上がり（もしくは立ち下がり）で出力が反転する。

【回路図（例）】

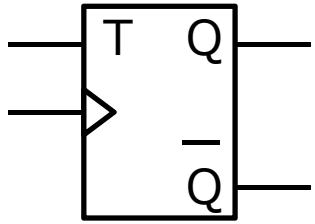


Dフリップフロップを利用

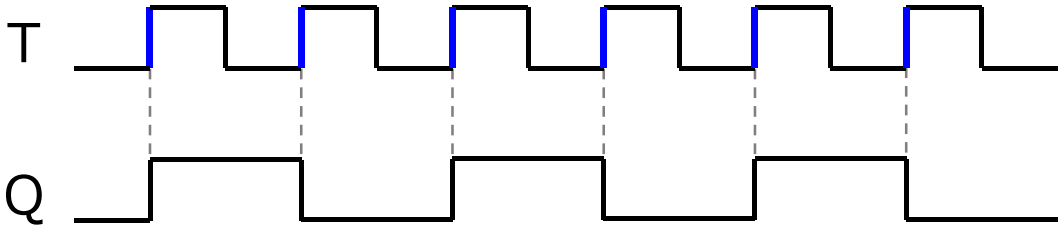


JKフリップフロップを利用

【シンボル】



【動作（タイミングチャート）】



基本的なデジタル回路
（フリップフロップを使った回路）

レジスタ

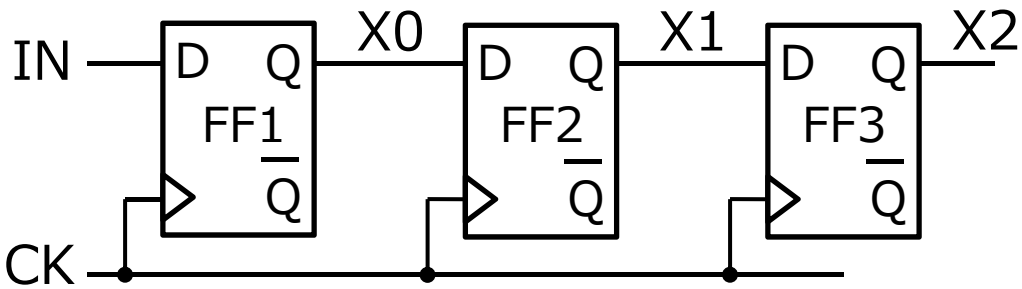
レジスタ（Register）はPCのプロセッサなどが内蔵する記憶回路。

制御装置、演算装置、実行ユニット等に直結し、動作速度が速く、比較的少量のものを指す。

一般的な論理回路においては、フリップフロップなどにより状態を保持する装置をレジスタと呼ぶ。

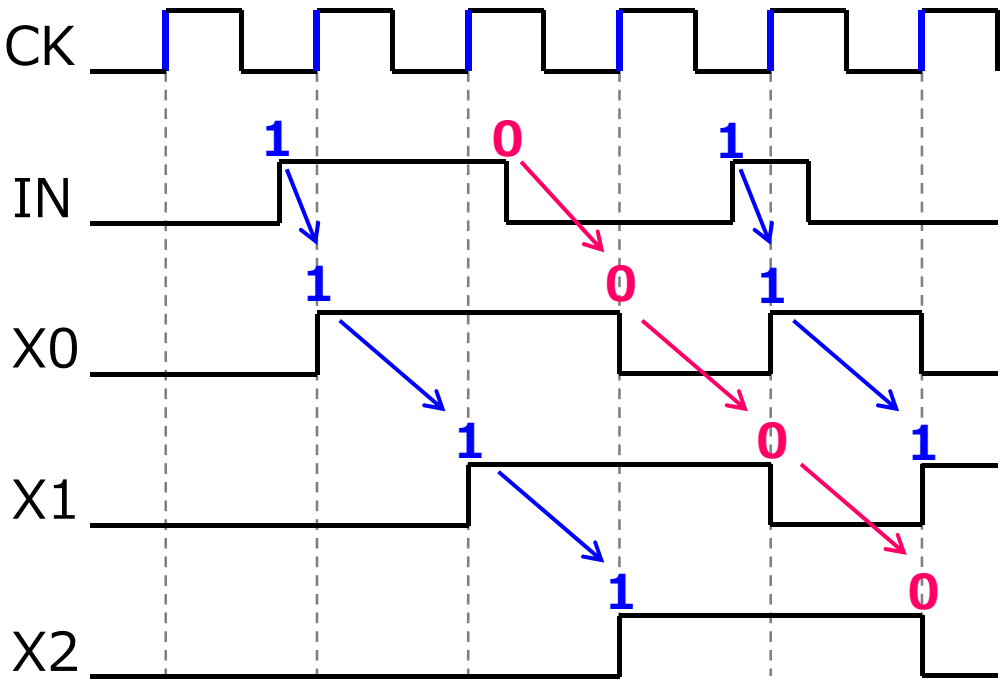
シフトレジスタ

【レジスタ - 回路図】



- FF1は、INを保持
- FF2は、FF1の出力Qを保持
- FF3は、FF2の出力Qを保持

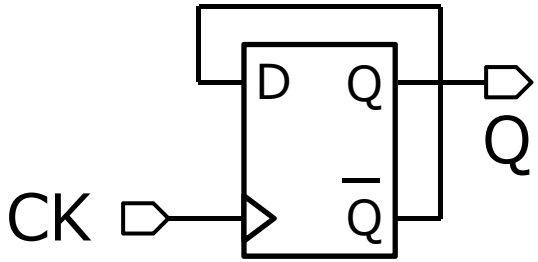
クロック(CK)に同期し、情報を1ビットずつ隣のフリップフロップに移動していくレジスタ。



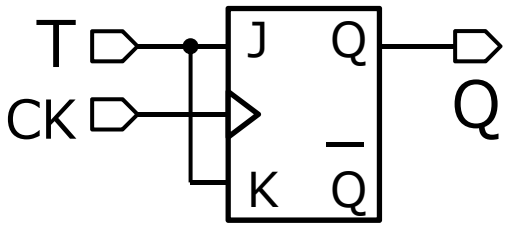
カウンタの要素回路

【回路】

T-FF（D-FFを使う場合は入力Dと反転出力を接続）
下図の回路はCKの立ち上がりで反転した値を保持する



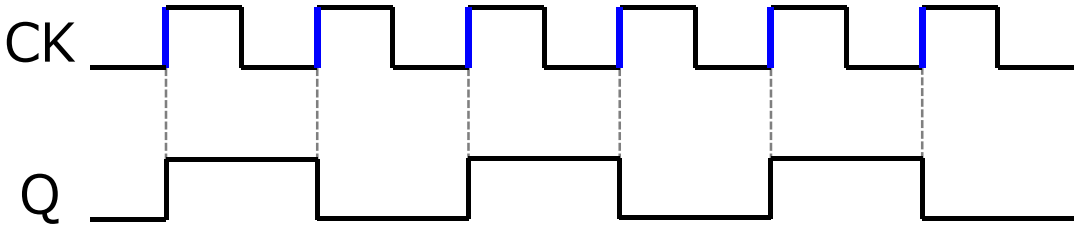
Dフリップフロップを利用



JKフリップフロップを利用

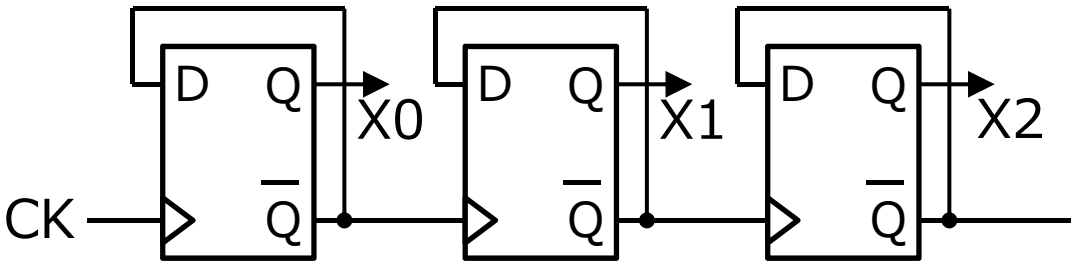
【回路動作】

D-FFはCKの立ち上がりで出力が変化、つまりQを反転させる



非同期カウンタ　アップカウンタ

【カウンタ - 回路図】



デジタル回路で扱う1または0の数をカウントする回路をカウンタと呼ぶ。
X2,X1,X0が出力（二進数）。

【回路動作】

・1段目のD-FF、

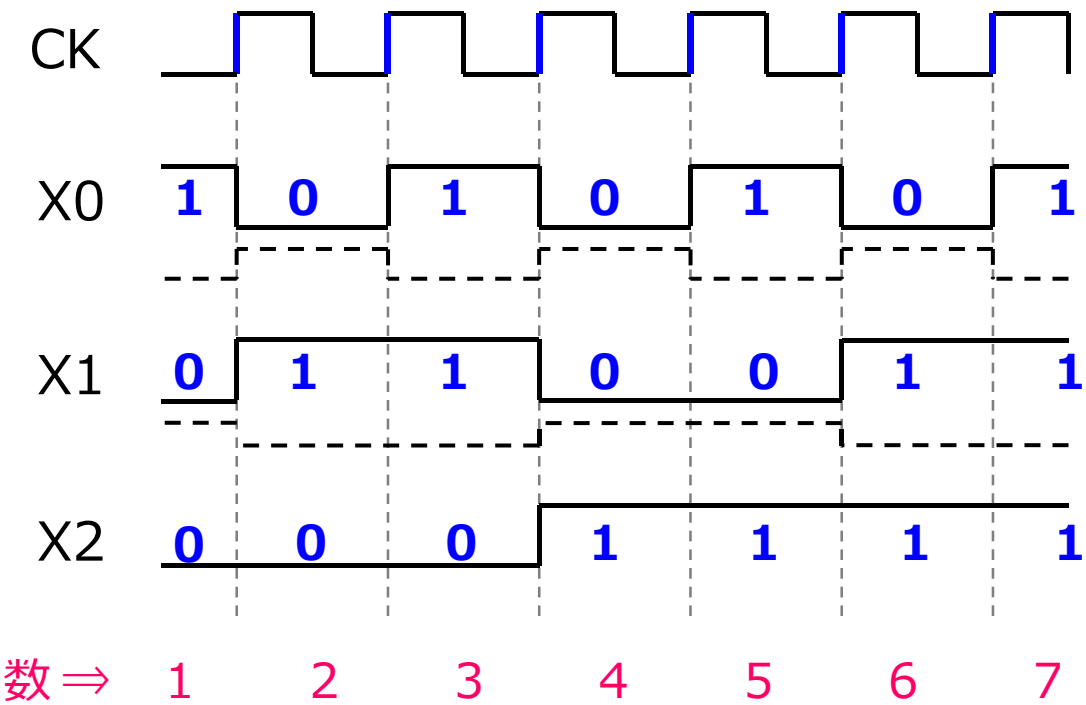
CKの立ち上がりで反転

・2段目のD-FF、

$\overline{X0}$ $\overline{(Q)}$の立ち上がりで反転

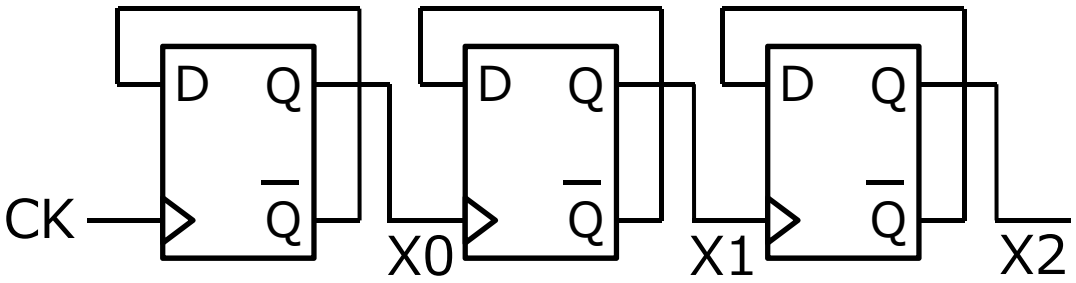
・3段目のD-FF、

$\overline{X1}$ $\overline{(Q)}$の立ち上がりで反転



非同期カウンタ　ダウンカウンタ

【カウンタ - 回路図】



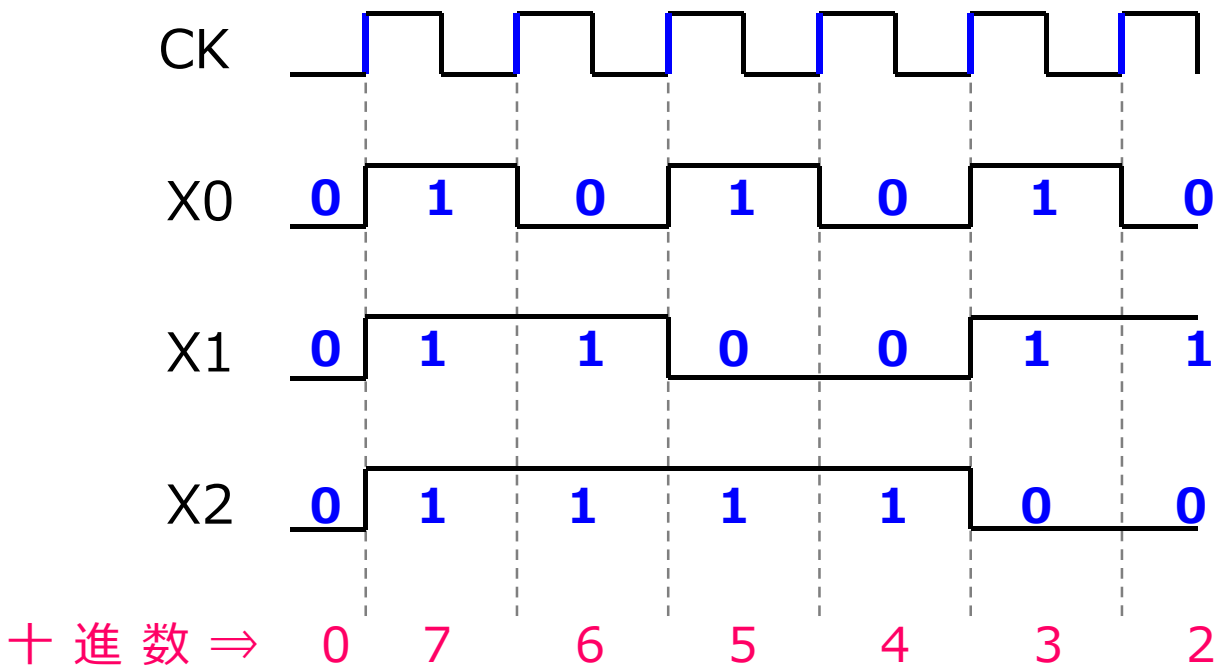
デジタル回路で扱う1または0の数をカウントする回路をカウンタと呼ぶ。
X2,X1,X0が出力（二進数）。

【回路動作】

・1段目のD-FF、
　CKの立ち上がりで反転

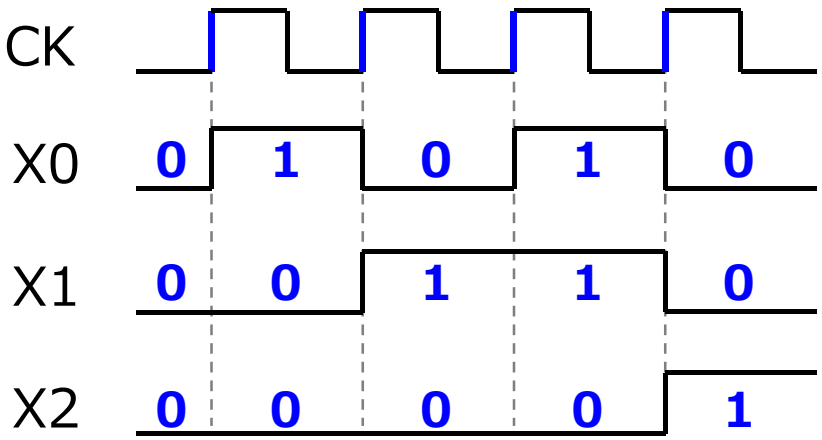
・2段目のD-FF、
　X0の立ち上がりで反転

・3段目のD-FF、
　X1の立ち上がりで反転

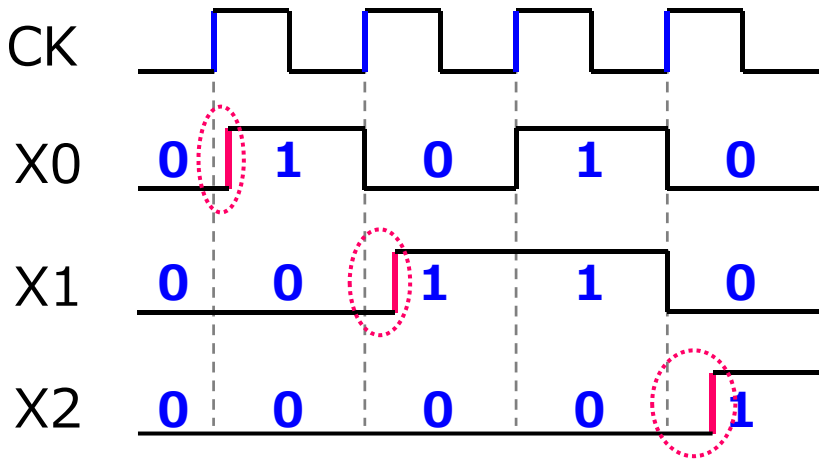


非同期カウンタと同期カウンタ

【遅延なしの非同期カウンタ】

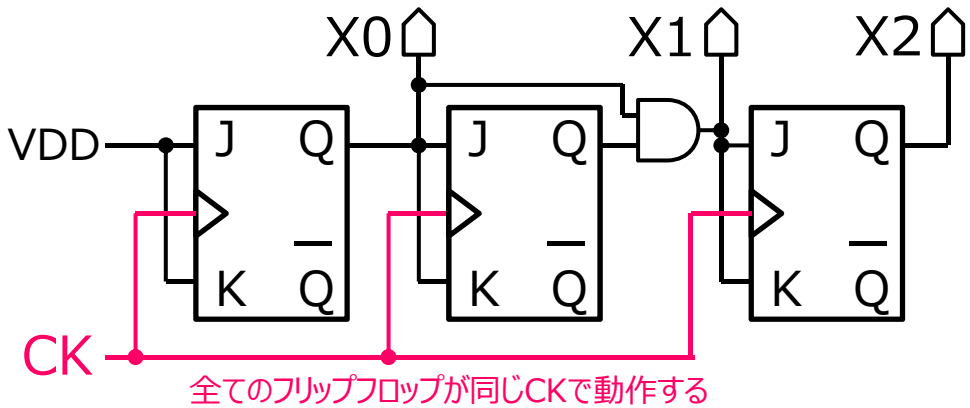


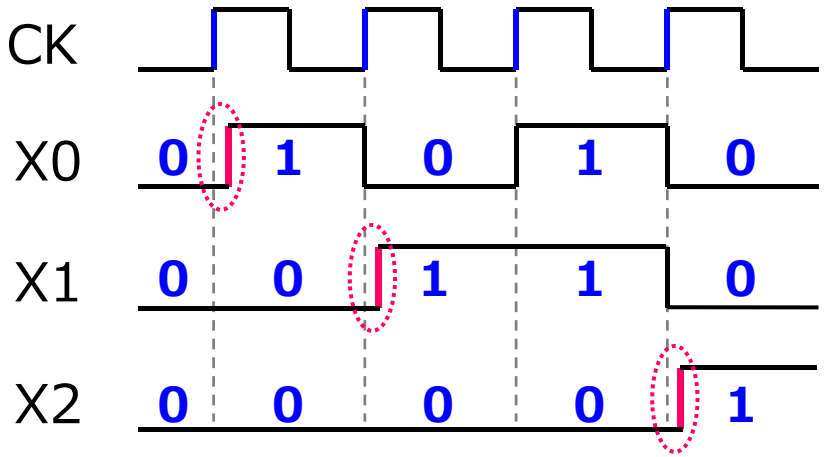
【遅延ありの非同期カウンタ(現実)】



段数が増えると遅延が大きくなる

【同期カウンタ】 段数が増えても遅延が一定





基本的な演算回路

加算器

加算器（かさんき：Adder）は加算を行う演算回路のこと。ここではデジタル加算回路について説明する。アナログ加算回路にはオペアンプを使った電圧加算器や電流源を使った電流加算器等が存在する。

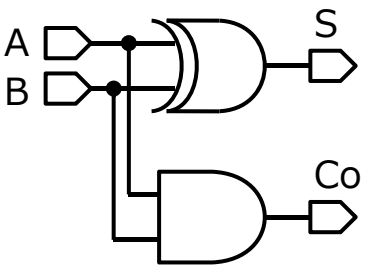
【半加算器（Half adder）】

半加算器（はんかさんき）は、2進数の同じ桁どうしを加算する回路。桁上がりは桁上げ出力（Carry out：Co）に出力する。

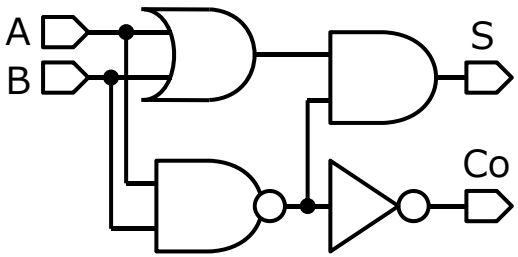
入力		出力	
A	B	S	Co
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

半加算器の真理値表

＜回路構成例＞



XORを使った場合

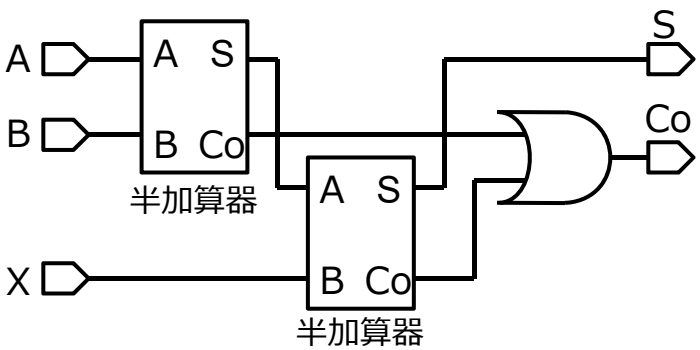


XORを使わない場合

【全加算器（Full adder）】

全加算器（ぜんかさんき）は、半加算器に入力を1つ追加し、下位bitからの繰り上げ計算をできるようにした回路。3入力の2進数を加算する回路と考えても良い。

入力			出力	
A	B	Ci	S	Co
0	0	0	0	0
0	1	0	1	0
1	0	0	1	0
1	1	0	0	1
0	0	1	0	1
0	1	1	1	0
1	0	1	1	0
1	1	1	1	1

全加算器の真理値表

＜回路構成例＞