

専門基礎科目 - 必修科目

GB10804 論理回路

1

システム情報系情報工学域 山口佳樹

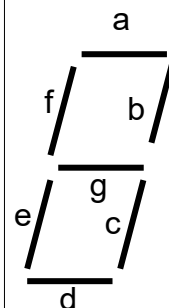
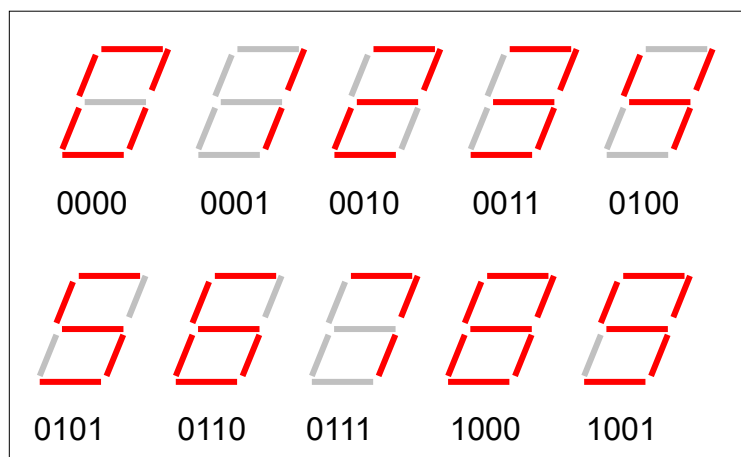
専門基礎科目 - 必修科目 (情報科学類)  
基礎科目 - 関連科目 (情報メディア創成学類)  
基礎科目 - 関連科目 (工学システム学類)

1

2

組み合わせ回路の例題 (教科書)

- 7セグメントの表示機を点灯させる回路



2

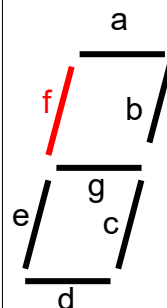
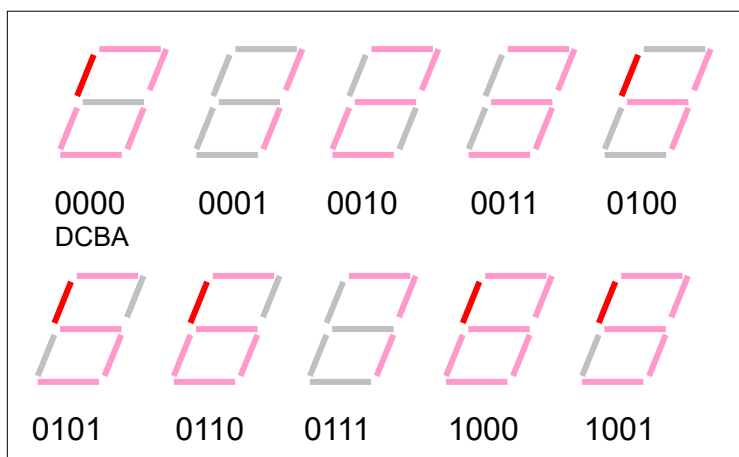
## 入出力関係

- 入力 D,C,B,A の 4ビットの信号
- 出力 a,b,c,d,e,f,g の7ビット
- 出力の信号のどれかは、入力信号の4ビットの値が決まれば、1意に決まる。
- $f = \text{func}(D,C,B,A)$

3

## 例題

- セグメント f について



4

5

## 例題

• セグメント f について

0000    0001    0010    0011    0100

DCBA

0101    0110    0111    1000    1001

101\*

11\*\*

は、  
don't care

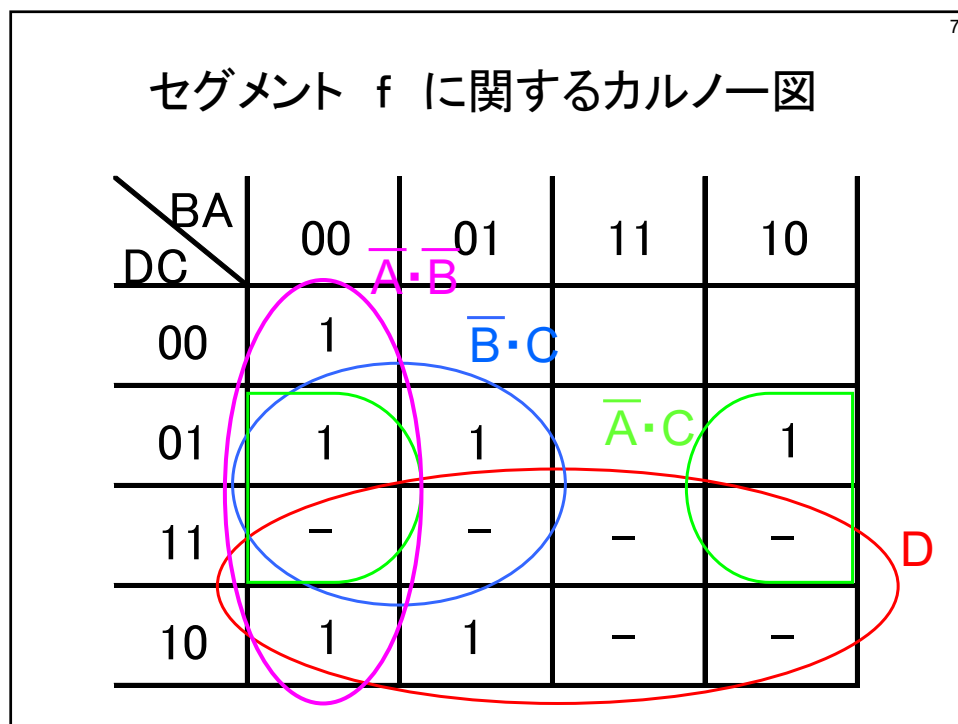
5

6

## セグメント f に関するカルノー図

BA DC	00	01	11	10
00				
01				
11				
10				

6



7

8

### 課題1 (Manabaに提出すること)

- 以下の 7セグを正しく表示するための論理式を示しなさい。式は、a~g の 7本となるはずである。

0000 DCBA    0001    0010    0011    0100

0101    0110    0111    1000    1001

8

## 課題2（Manabaに提出すること）

### 四捨五入回路を作成する

0 から 9 までの値を取るある乱数が 2 進数で入力されるとする。  
このとき、0 から 4 までの入力に対しては 0 を、5～9 までの入力  
に対しては 1 を出力する四捨五入回路を設計しなさい。

ここで、2 進数の入力値は 4 bit であり、上位より "DCBA" で  
表されるとする（例：7 が入力 → {DCBA}={0111}）。  
このとき、カルノー図はスライドのフォーマットを利用して書きなさい。

そして、得られたカルノー図を基に、入力に対する出力の論理式  
も示しなさい。

9

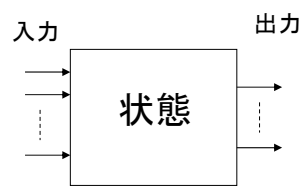
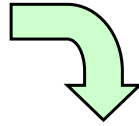
### 0～9 までの入力に対するカルノー図 (Don't Care を使用して構いません)

BA \ DC	00	01	11	10
00				
01				
11				
10				

10

## 順序回路

- 入力信号と回路の状態で、出力が決まる
- 状態を記憶するものが必要



フリップフロップ(FF)  
や  
ラッチ

11

## 「ラッチ」と「フリップフロップ」

ある状態を記憶しておく機能を持つ素子

### A) 「Latch(ラッチ)」

- 内部状態の変化は「G」入力で制御
- SR ラッチ、D ラッチ、など

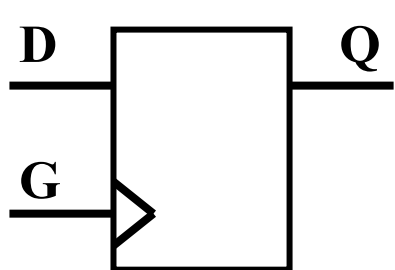
### B) 「Flip-Flop(フリップフロップ)」

- 内部状態の変化は「clock」と同期
- D-FF、JK-FF、T-FF、など

12

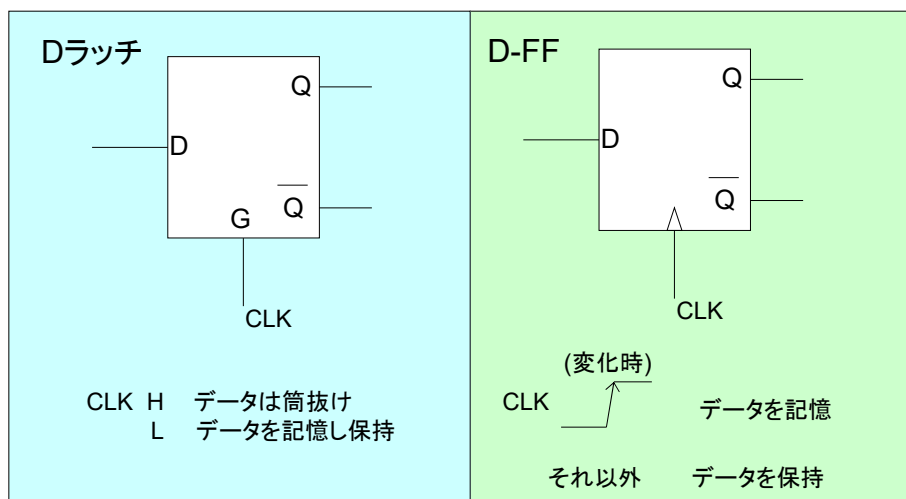
## 「ラッチ」と「フリップフロップ」

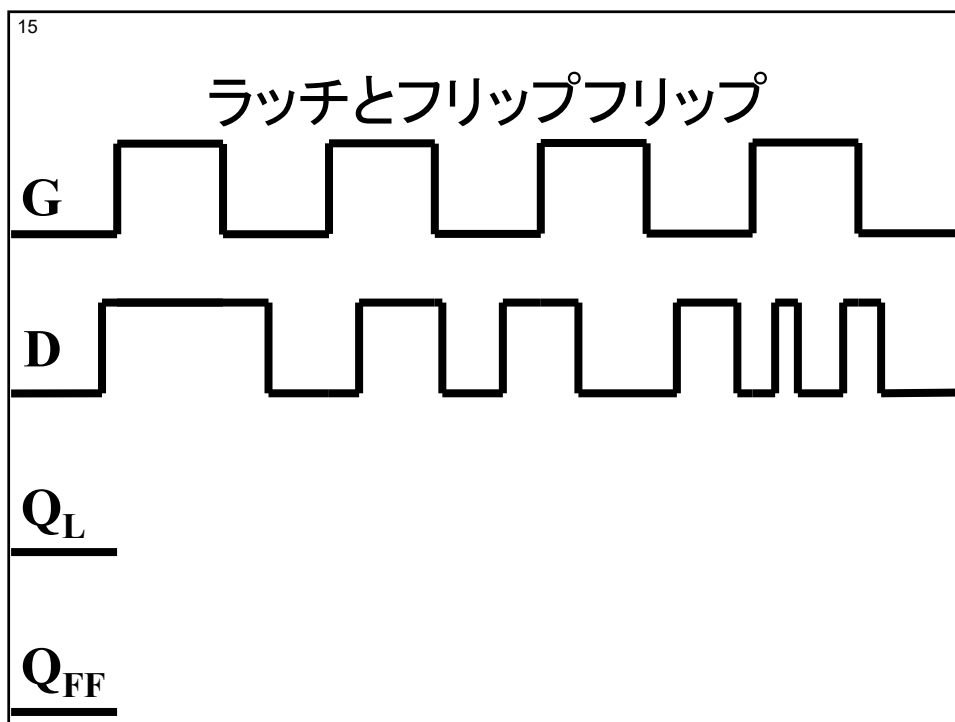
### ラッチとフリップフロップの違いについて (例: D-Latch と D-FF)



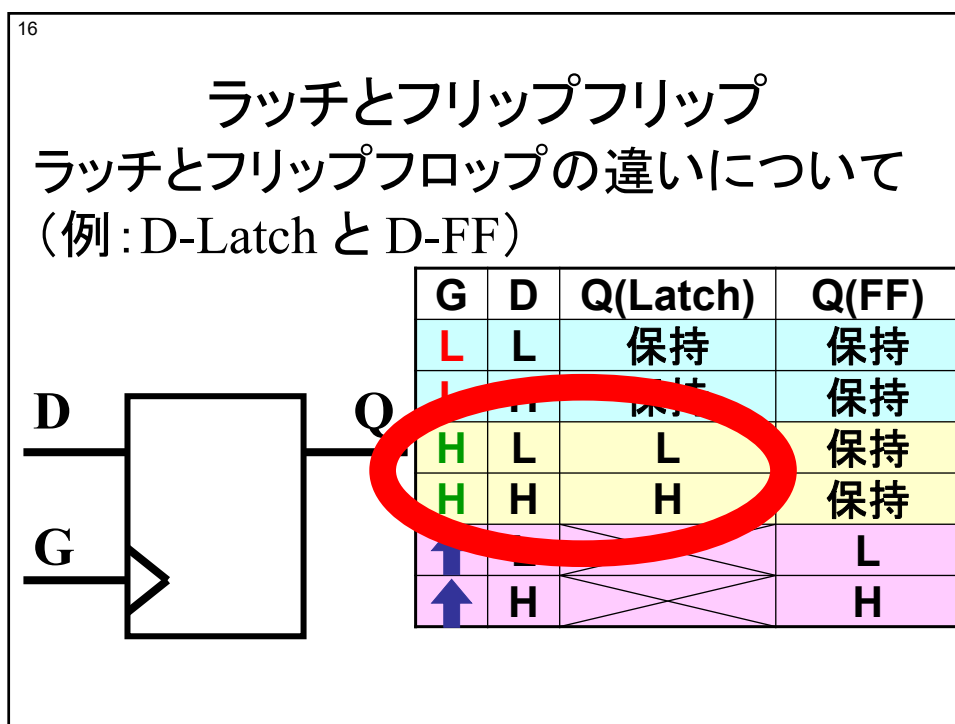
G	D	Q(Latch)	Q(FF)
L	L	保持	保持
L	H	保持	保持
H	L	L	保持
H	H	H	保持
↑	L		L
↑	H		H

## DラッチとD-フリップフロップ(FF)



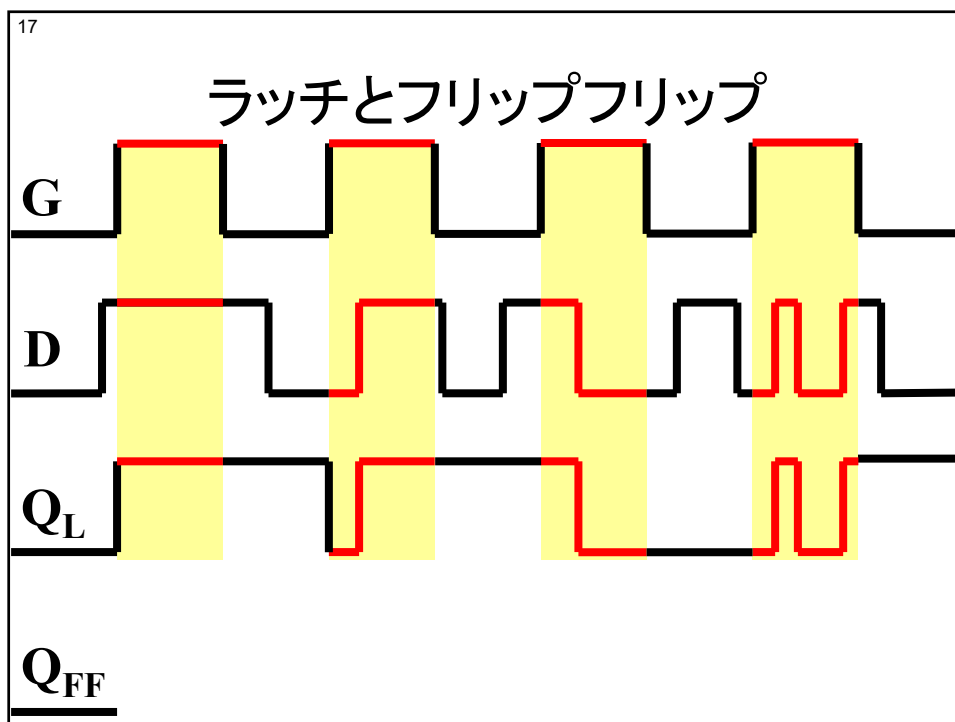


15



16





17

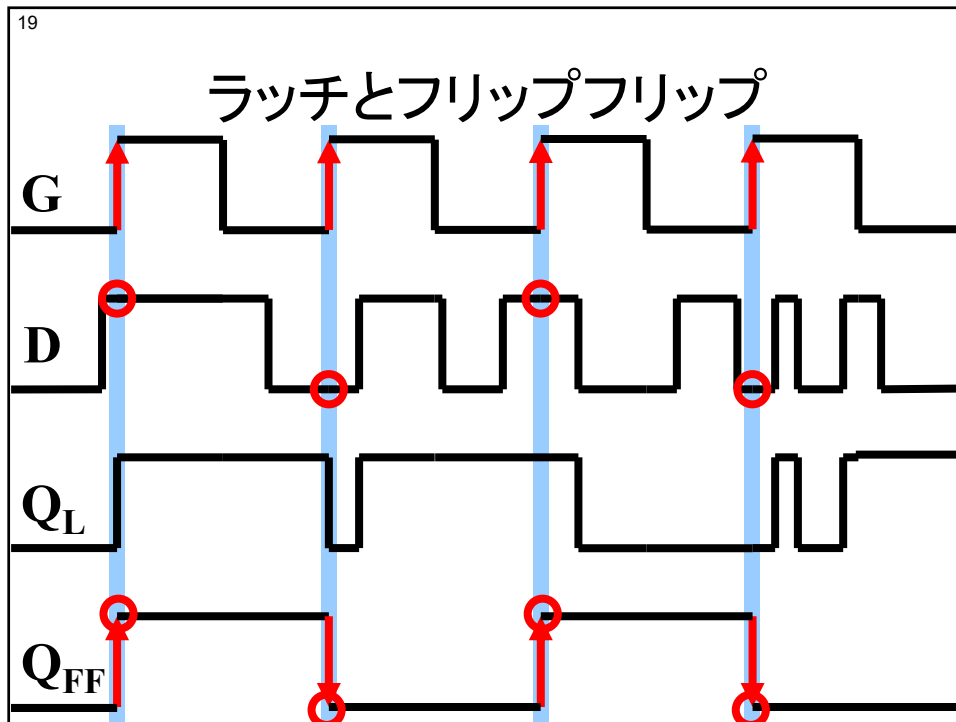
18

### ラッチとフリップフリップ

ラッチとフリップフロップの違いについて  
(例: D-Latch と D-FF)

G	D	Q(Latch)	Q(FF)
L	L	保持	保持
L	H	保持	保持
H	L	L	保持
H	H	H	保持
↑	L		L
↑	H		H

18



19

20

### (まとめ)ラッチとフリップフリップ

同じ記憶素子でも保存の仕方が異なる

A) Latch (ラッチ)

- ・ 内部状態の変化は **G 入力**で制御
- ・ SR ラッチ、D ラッチ、など

B) Flip-Flop (フリップフロップ)

- ・ 内部状態の変化は **clock と同期**
- ・ D-FF、JK-FF、T-FF、など

20

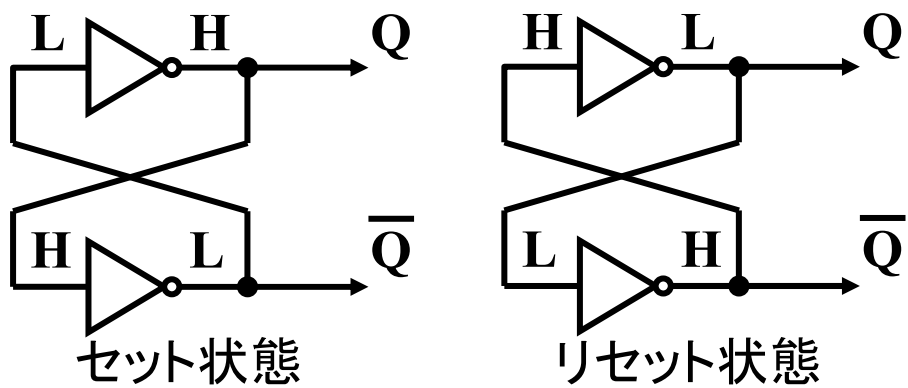
21

論理回路から電子回路へ  
記憶をさせるためには……

21

22

論理回路から電子回路へ  
記憶をさせるためには……  
⇒ 情報をループさせる必要がある。

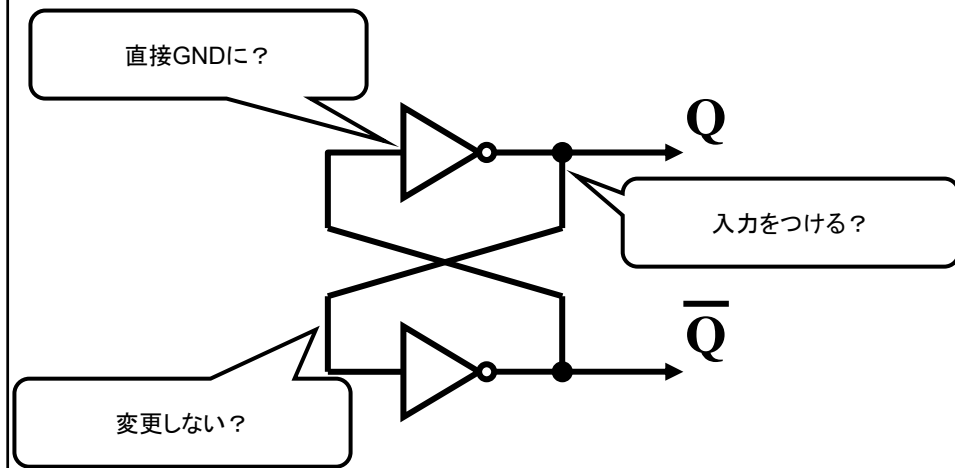


22

23

## 論理回路から電子回路へ

(問題) 入力がない  $\Rightarrow$  記憶を変更できない

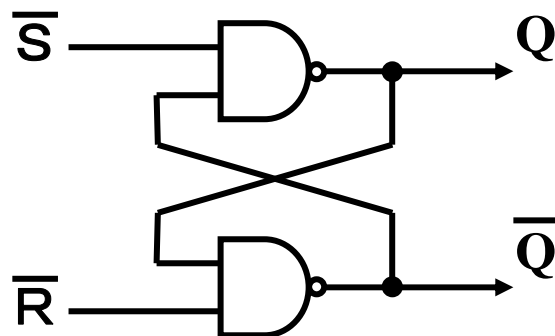


23

24

## 「SRラッチ」

(解決) 入力をつける  $\Rightarrow$   $\bar{S}\bar{R}$  ラッチの導入

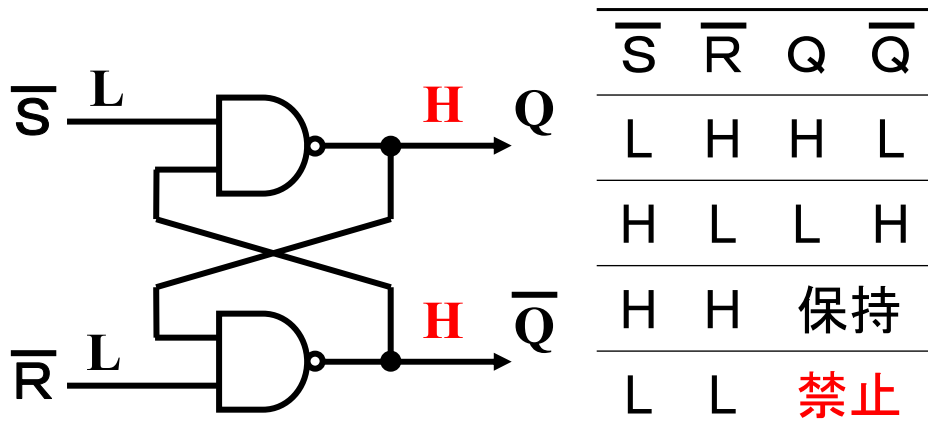


24

25

## 「 $\overline{S}\overline{R}$ ラッチ」

(解決) 入力をつける  $\Rightarrow \overline{S}\overline{R}$ ラッチの導入

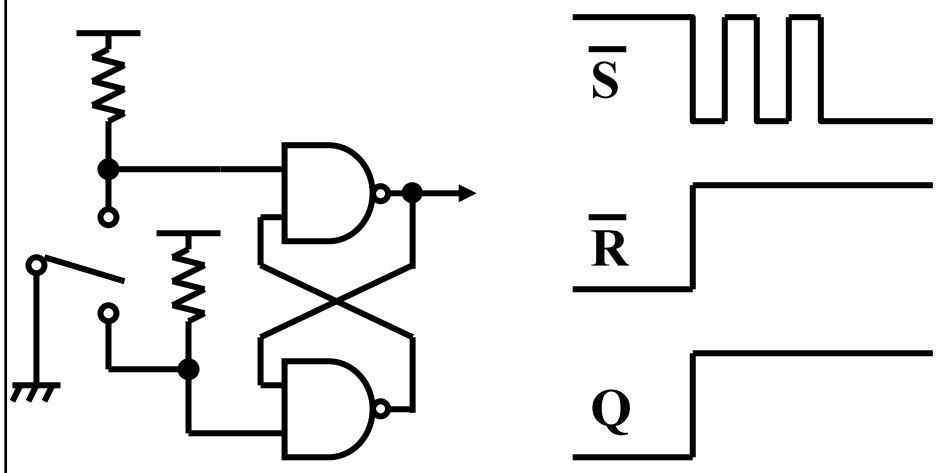


25

26

## SRラッチの応用

「チャタリング」の防止回路

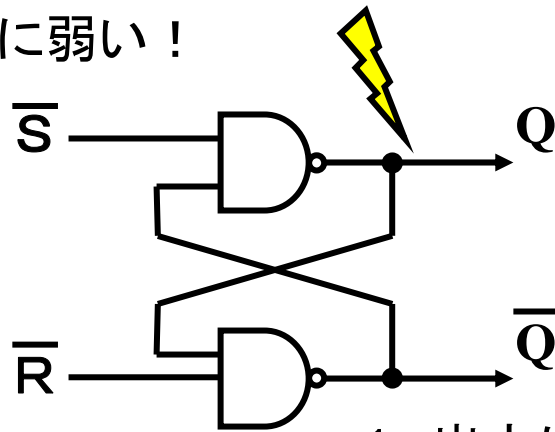


26

27

## $\overline{S}\overline{R}$ ラッチの問題

ノイズに弱い！



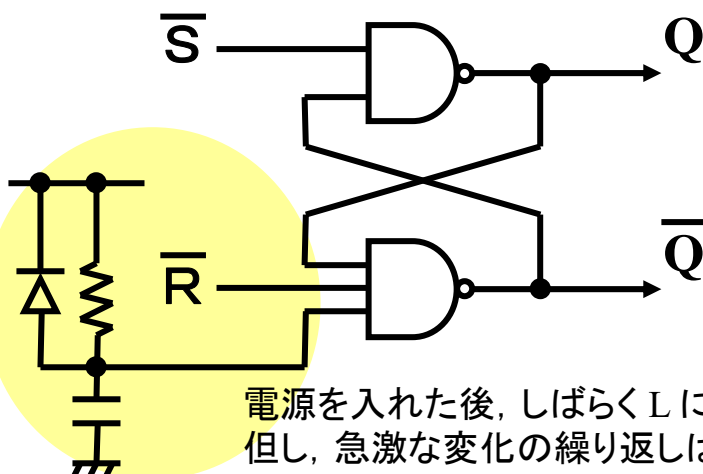
1. 出力は短くする
2. 電源投入時？

27

28

## $\overline{S}\overline{R}$ ラッチの問題

【電源投入時】パワーオンリセット機能

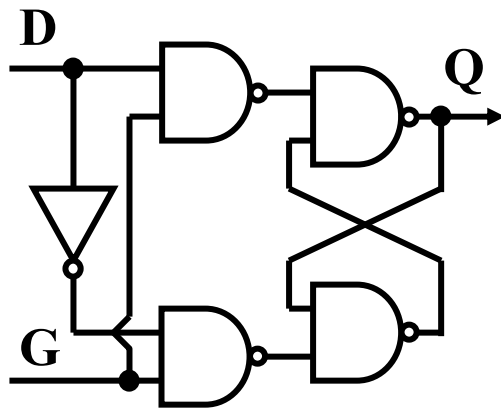


電源を入れた後、しばらくLに固定しておく。  
但し、急激な変化の繰り返しは×。

28

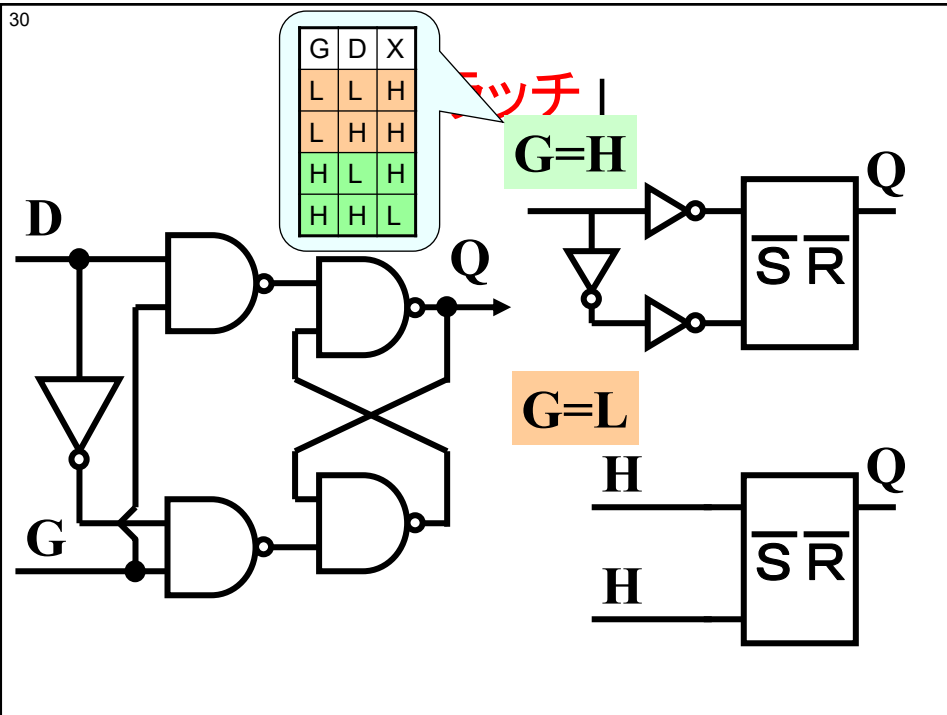
29

# 「D-ラッチ」



29

30

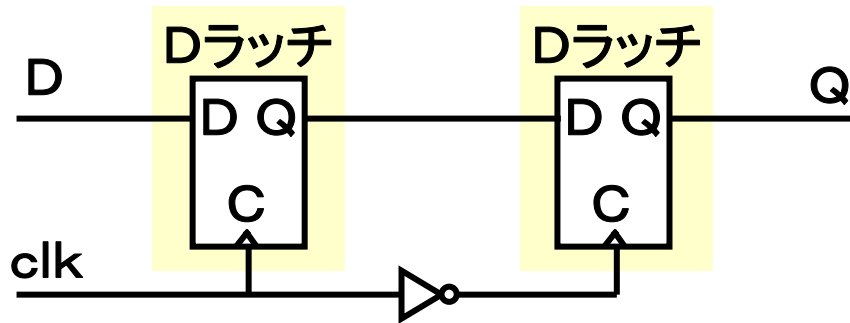


30

31

## 「D-フリップフロップ」

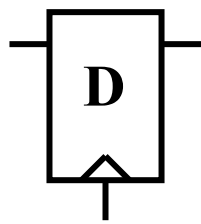
Dラッチ2個がベースになっている  
(立下り感知型Dフリップフロップ)



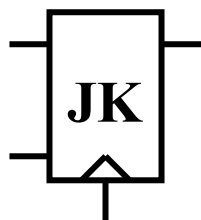
31

32

## 色々なフリップフロップ



D	CLK	Q
L	↑	L
H		H
L	他	保持
H		



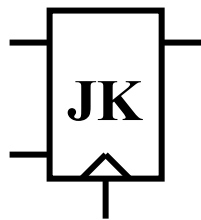
J	K	CLK	Q
H	L	↑	H
L	H		L
H	H		逆保持
L	L		保持

32

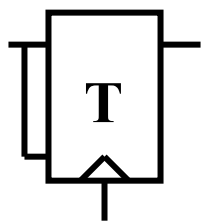


33

## 色々なフリップフロップ



J	K	CLK	Q
H	L	↑	H
L	H		L
H	H		逆保持
L	L		保持



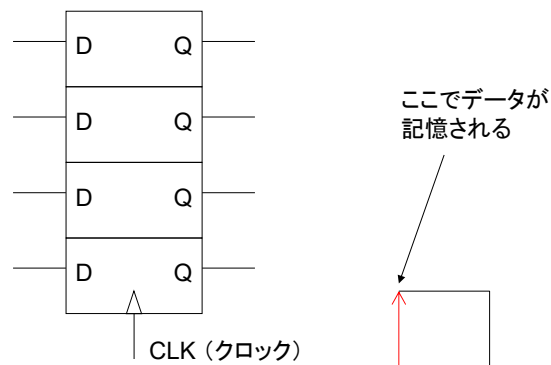
T	CLK	Q
H	↑	逆保持
L		保持

33

## D-FFの応用(1)

- レジスタ
  - データを記憶する

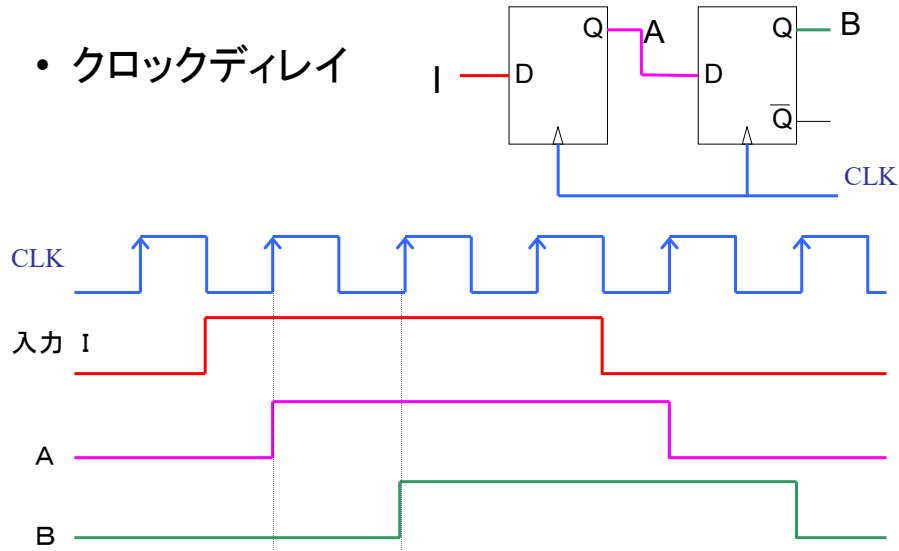
4ビットのレジスタ



34

## D-FFの応用(2)

- クロックディレイ



35

## 順序回路の基本

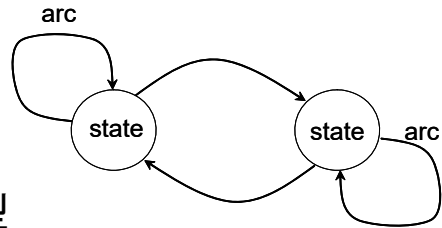
- 有限状態機械(Finite State Machines)によって記述
  - 有限の(数えられる)状態とその変化で構成されるもの
- 有限状態機械の記述法
  - 状態遷移図
  - ハードウェア記述言語
  - ブール代数

36

## 状態遷移図

- 取りうるべき状態と状態から状態への変化を表現したグラフ

- 状態 (state)
- 方向のある線(arc)

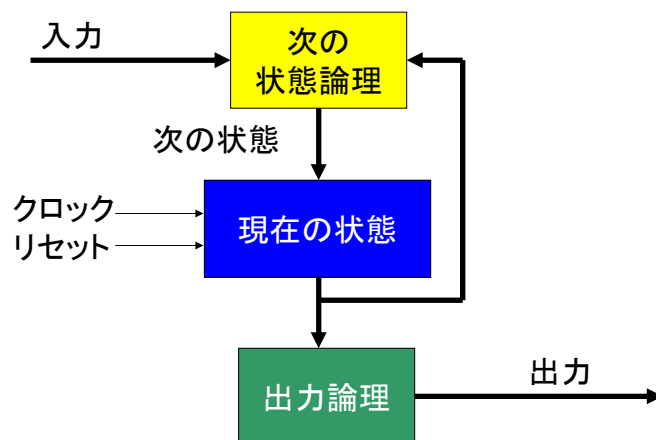


- ムーア型とミーリー型

37

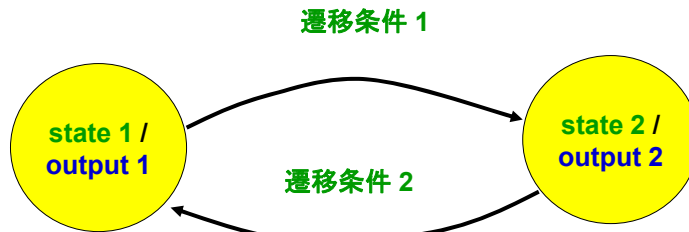
## ムーア (Moore) 型

- 出力は、現在の状態のみに依存する



38

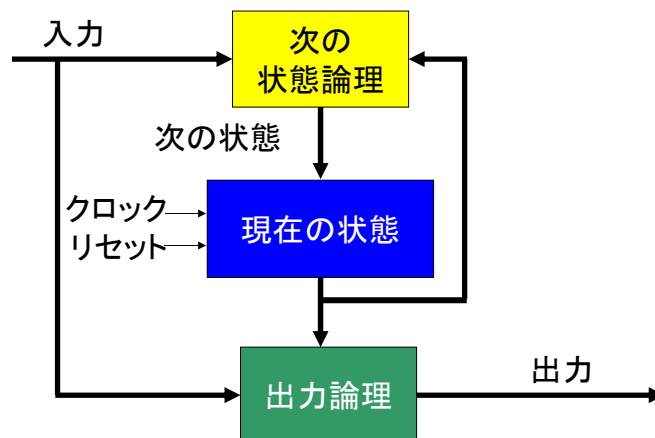
## ムーア (Moore) 型



39

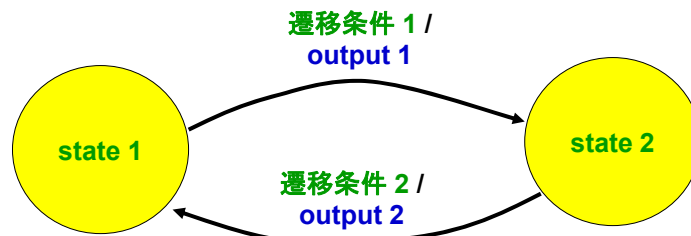
## ミーリー (Mealy) 型

- 出力は、現在の状態と入力(状態変化)に依存する



40

## ミーリー (Mealy) 型



41

## ムーア型とミーリー型の比較

- 表現能力は同じ(互いに変換できる)
- ミーリー型は、ムーア型に比べて、入力の変化に対してすばやく出力を出すことができる
  - ムーア型は、出力の反応が1クロック遅れる
  - クロックからの遅れはムーア型の方が早い(クリティカルパスに対して有利)
- 一般的に、ミーリー型の方がより少ない状態で記述することができる

ミーリー型を使うことが一般的

42

## 順序回路の設計 (フリップフロップ プログラミング)

1. 状態遷移図(ミーリーグラフ)
  - 必要があれば、状態遷移図の簡単化
2. 状態遷移表の作成
3. 状態に2進数を割り当てる
4. 組み合わせ論理回路による設計
  - 次の状態
  - 出力

43

## 簡単な例による設計

- 連続して1または0が、入力されると出力が1(それ以外は0)となる順序回路を設計する
- 組み合わせ回路では実現できない
  - 前の入力を覚えていなければならない

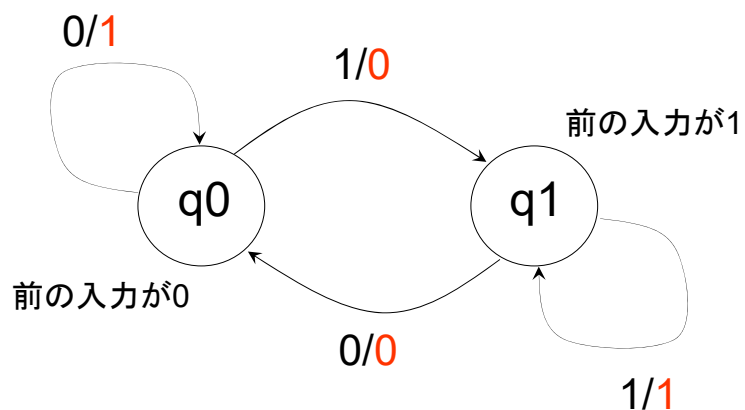
44

## 状態はいくつ必要か？

- 2つの状態が必要
  - 前の入力が、0か1か
- q0: 前の状態が0  
q1: 前の状態が1

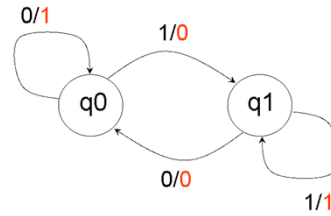
45

## 状態遷移図 (ミーリー型)



46

## 状態遷移表



入力	現在の状態	次の状態	出力
S	C	N	Y
0	0	0	1
0	1	0	0
1	0	1	0
1	1	1	1

47

## 次の状態 Nの カルノー図

S \ C	0	1
	0	1
0	0	0
1	1	1

$$N = S$$

48



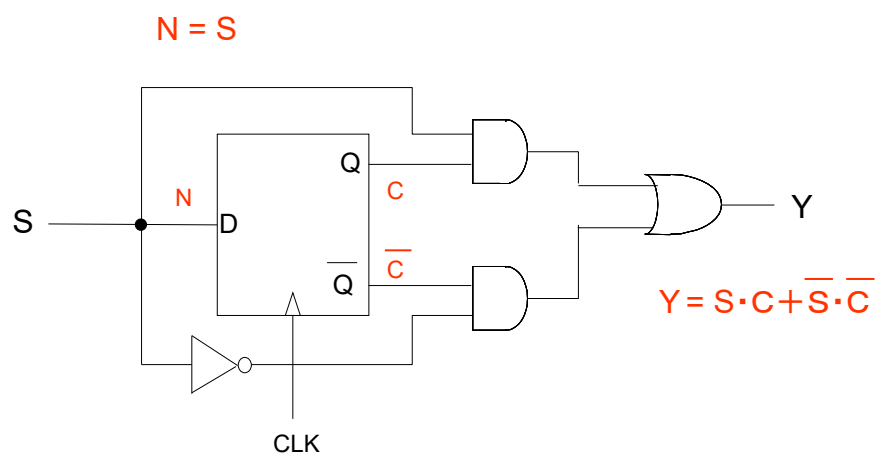
## 出力 Yの カルノー図

$S \backslash C$	0	1
0	1	0
1	0	1

$$Y = S \cdot C + \overline{S} \cdot \overline{C}$$

49

## 回路図



50

## 例題

- $\rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 3 \rightarrow \dots$  と数えるような**カウンタ**を、フリップフロッププログラミングで設計せよ。
  - ただし、サイコロカウンタと同様に、入力信号  $S = H(1)$  の時は動作し、 $L(0)$  の時は止まっている(同じ値でいる)ものとする。
  - (ヒント: 状態は5つしかないが、状態を表すのに3bit必要。カルノー図にはdon't careが出て来る)
  - 解答は、状態を表す3bitのそれぞれの論理式を導くまでも構いませんが、余裕のある人は回路図まで考えてみてください。

51

## カルノー図 現在の状態 (Current State) から次の状態 (Next State) を導く。

$N_2$					$N_1$					$N_0$				
$SC_{2^3}C_1C_0$	00	01	11	10	$SC_{2^3}C_1C_0$	00	01	11	10	$SC_{2^3}C_1C_0$	00	01	11	10
00					00					00				
01					01					01				
11					11					11				
10					10					10				

52