2021年5月11日,18日(火)

集積回路システム工学　第4、5回講義

デジタルCMOS回路の基礎

小林春夫
群馬大学大学院理工学府　電子情報部門
koba@gunma-u.ac.jp

下記から講義使用　pdfファイルをダウンロードしてください。
出席・講義感想もここから入力してください。
https://kobaweb.ei.st.gunma-u.ac.jp/lecture/lecture.html

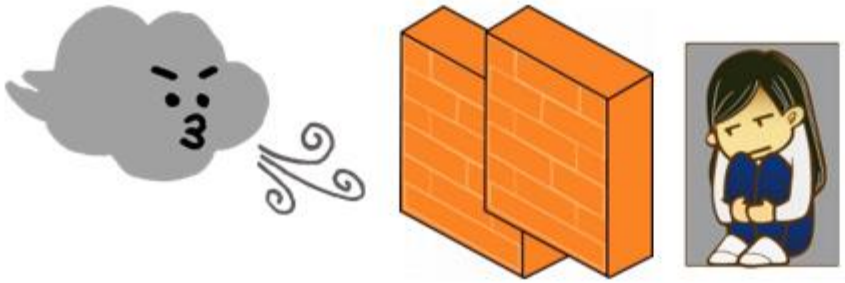
2020年3月31日

コロナウィルス影響で激変する環境に如何に対応すべきか

群馬大学　小林春夫

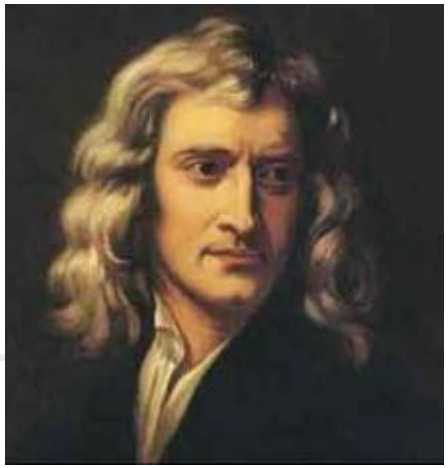
中国の諺（ことわざ）

風向きが変わるとき、
ある者は塀を建て、ある者は風車を作る
见风使舵（風向きをみて舵をとる）



Don't be afraid of change!

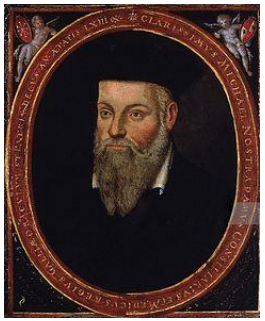
ニュートンの大発見



アイザック・ニュートンはケンブリッジ大学卒業の1665 年に、ペスト流行のため大学 が閉鎖され郷里に帰り1年半過ごす。

ここで3つの大理論の端緒を次々に発見する。

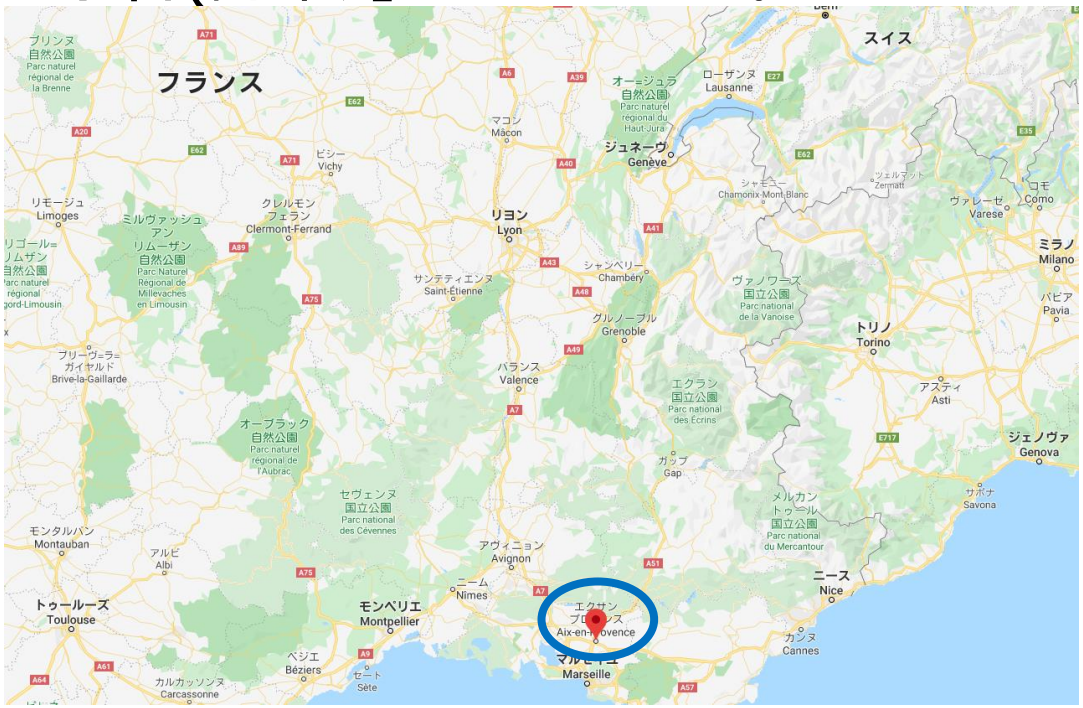
ルネッサンスの予言者 ペストから町を救う



ミシェル・ノストラダムス
1503-1566

ノストラダムスは医師、占星術師。

「大予言(詩集)」でも知られる。



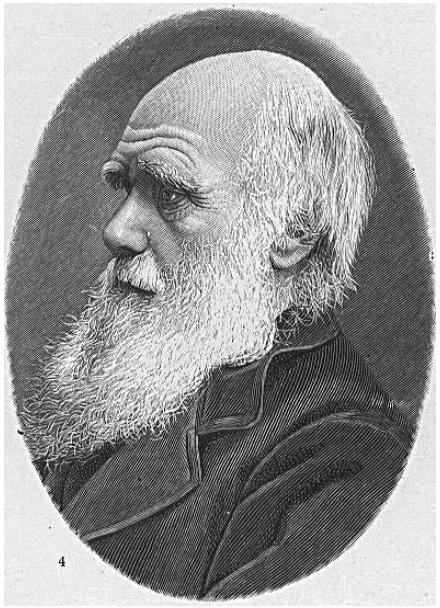
南仏の都市エクサン・プロバンスをペストから救う。

（近くの半導体関係の会社を20年以上前に訪問）

激変する技術･経済環境下で

Charles Robert Darwin

進化論



激変する環境下で生き残る生物。
強い者でもない
賢い者でもない。
変化する者だけが生き残る。

米国からなぜ新しい技術が生まれるか

- ソニー　盛田昭夫氏

  米国では different であることを好む

  日本では uniform であることを好む

  米国は多民族国家

- なぜ生物にオスとメス、男と女

  個体が全て同じならウィルスに侵される。

  異なる個体はウィルスに強い。

「勝つ」ことより「負けない」こと

双六の上手といひし人に, その手立を問ひ侍りしかば

「勝たんと打つべからず。負けじと打つべきなり。」

（吉田兼好 徒然草）

「勝ちに不思議の勝ちあり。

負けに不思議の負けなし。」

（プロ野球　野村克也氏）

危険と好機

第35代　米国大統領
John F. Kennedy

Crisis（危機）という言葉は
二つの漢字でできている。
ひとつは危険、もうひとつは好機である。

When written in Chinese,
the word ‘crisis’ is composed of two characters.
One represents danger and
the other represents opportunity.

チャンスとピンチ

「若手はチャンスに強い。
ベテランはピンチに強い。」

（将棋プロ棋士　羽生善治氏）

疾風に勁草を知る

「（将棋の）弱い人ほど結論を早く出したがる」

（大山康晴　将棋１５世名人）

簡単に結論を出さない。

「人間力」を鍛える

「死と向かいあった捕虜の世界では、皆平等である。
実社会で威張っていた人物ほど、
極限状態に置かれたら　だらしないのを
ずいぶん見たものだ。
いまだに、肩書きや学歴を鼻にかける人間が
信用できないのは、ことのときのあまりにも
大きな落差を知っているからである。」

（シベリア抑留経験、再建王　坪内寿夫）

最後に

ウィルスの影響が大きい中で

偉人達の言動に

「一灯をさげて暗夜を行く。

暗夜を憂うなかれ、一灯を頼め。」

（国学者　佐藤一斎 「言志晩録」）

MWE基礎講座

2010年12月10日

デジタル・アシスト・アナログ技術のための デジタルCMOS回路設計　再入門

群馬大学大学院 工学研究科 電気電子工学専攻

小林春夫

k_haruo@el.gunma-u.ac.jp

内　容

- セミナーの目的・目標
- デジタル回路の基礎
- スイッチレベル　デジタルCMOS回路
- デジタルCMOS回路の性能（消費電力、スピード）
- 同期回路設計とカウンタ回路
- 加算器、ビットシフト、乗算器
- 事例１：　SAR ADC+分散型積和演算回路
- 事例２：　自己校正機能をもったTDC回路
- 事例３：　冗長アルゴリズムを用いたSAR ADC
- 最後に

内　容

- セミナーの目的・目標
- デジタル回路の基礎
- スイッチレベル　デジタルCMOS回路
- デジタルCMOS回路の性能（消費電力、スピード）
- 同期回路設計とカウンタ回路
- 加算器、ビットシフト、乗算器
- 事例1：　SAR ADC+分散型積和演算回路
- 事例2：　自己校正機能をもったTDC回路
- 事例3：　冗長アルゴリズムを用いたSAR ADC
- 最後に

セミナーの目的・目標

- アナログRF回路設計では
  デジタルアシスト・アナログ技術が重要
- 比較的小規模のCMOSデジタル回路
  設計の基本をレビューする
- 3つの事例を紹介する

内　容

- セミナーの目的・目標
- デジタル回路の基礎
- スイッチレベル　デジタルCMOS回路
- デジタルCMOS回路の性能（消費電力、スピード）
- 同期回路設計とカウンタ回路
- 加算器、ビットシフト、乗算器
- 事例1：　SAR ADC+分散型積和演算回路
- 事例2：　自己校正機能をもったTDC回路
- 事例3：　冗長アルゴリズムを用いたSAR ADC
- 最後に

アナログ信号とデジタル信号

アナログ信号

連続的な信号

例：　自然界の信号（音声、電波）、アナログ時計

「坂道」

デジタル信号

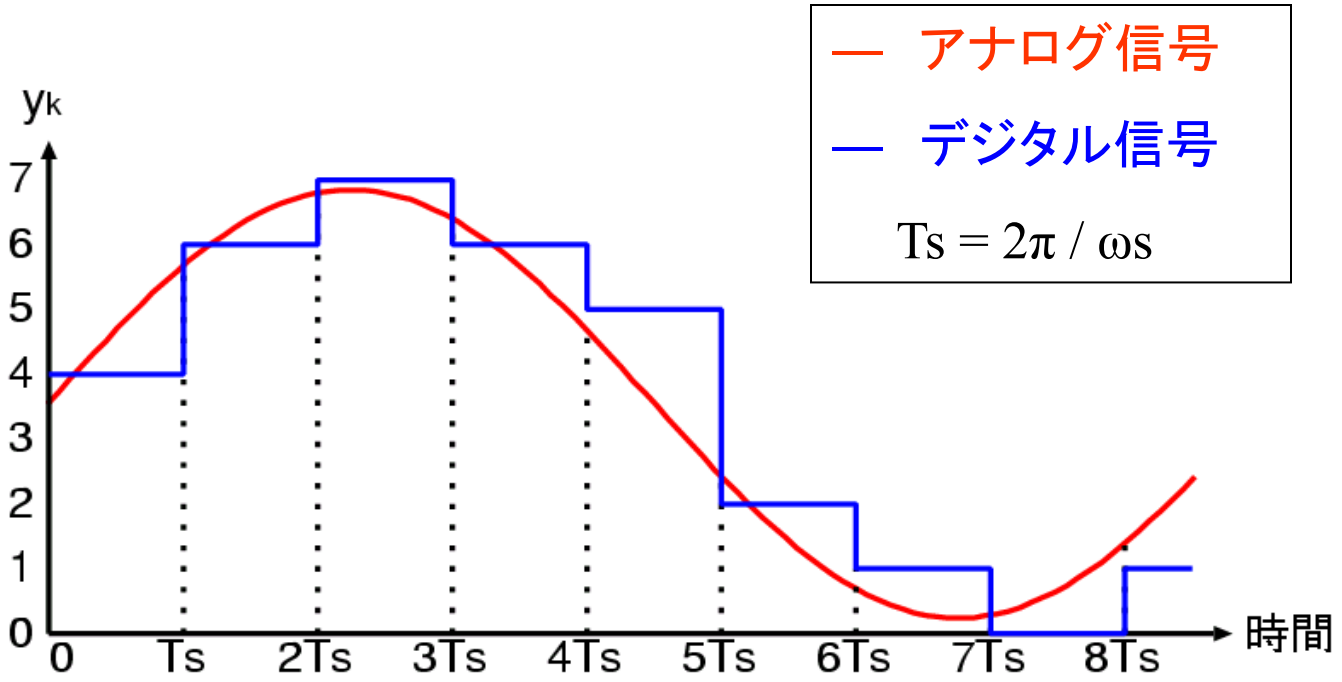
離散的・数値で表現された信号

例：コンピュータ内での２進数で表現された信号

デジタル時計

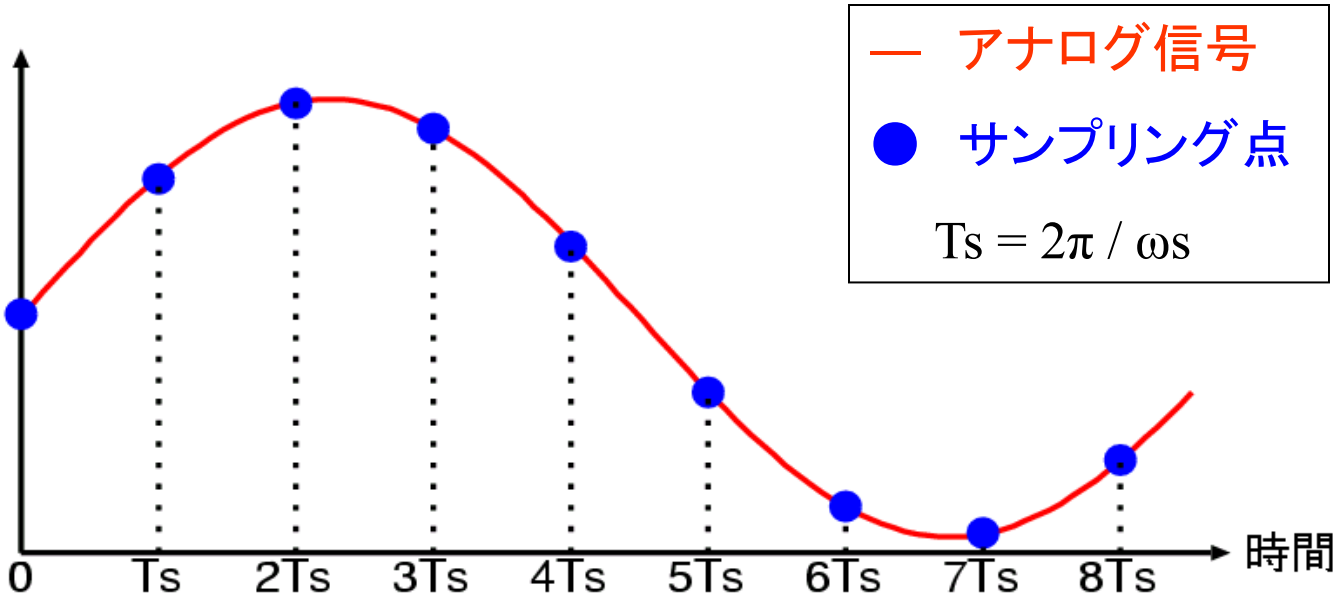
「階段」

デジタル信号の特徴（1）
空間の量子化（信号レベルの数値化）



デジタル信号はアナログ信号レベルを
四捨五入（または切り捨て）

デジタル信号の特徴（1）
時間の量子化（サンプリング）



一定時間間隔のデータを取り、間のデータは捨ててしまう。

デジタル回路と2進数

- 人間はなぜ10進数を使うか？

  → 手の指が10本あるから。

- デジタルではなぜ2進数を使うか？

  → 2つの状態は技術的に容易かつ安定して実現可能。

例：　電圧の高いと低い

　　電流の流れる状態と流れない状態

　　パルスのあるとなし。

１０進数と２進数

10進	２進	10進	２進
0	0000	8	1000
1	0001	9	1001
2	0010	10	1010
3	0011	11	1011
4	0100	12	1100
5	0101	13	1101
6	0110	14	1110
7	0111	15	1111

例　2進数　1011　を
　１０進数に変換
1x2x2x2+0x2x2+1x2+1
=11

2進数

$2^0 = 1$

$2^1 = 2$

$2^2 = 4$

$2^3 = 8$

$2^4 = 16$

$2^5 = 32$

$2^6 = 64$

$2^7 = 128$

$2^8 = 256$

$2^9 = 512$

$2^{10} = 1{,}024 = 1K$

(参考　1,000=1k)

１６進数、８進数とデジタル

10進	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
8進	0	1	2	3	4	5	6	7	10	11	12	13	14	15	16	17	20	21	22	23	24
16進	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14

- 人間はなぜ１０進数を使うか？

⇒ 手の指が１０本あるから。

- デジタルコンピュータは２進数が基本。

ではなぜ１６進数、８進数を使うか？

⇒ ２進数と１６進数、８進数は相性がよいから。

8進数と2進数の変換

8進	2進 (4 2 1)
0	0 0 0
1	0 0 1
2	0 1 0
3	0 1 1
4	1 0 0
5	1 0 1
6	1 1 0
7	1 1 1

例　8進4桁　3724

●10進に変換

3x8x8x8+7x8x8+2x8+4

計算が必要

●2進に変換

011 111 010 100

左表から機械的に得られる

１６進数と2進数の変換

16進	2進	16進	2進
0	0000	8	1000
1	0001	9	1001
2	0010	A	1010
3	0011	B	1011
4	0100	C	1100
5	0101	D	1101
6	0110	E	1110
7	0111	F	1111

例　16進で3桁
　A46
2進数に変換
1010 0100 0110
左表から機械的
に得られる

4ビットの場合

負の数の表現（2の補数）

符号無 ud	符号付 sd	2進 b3 b2 b1 b0
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111

符号無 ud	符号付 sd	2進 b3 b2 b1 b0
8	-8	1000
9	-7	1001
10	-6	1010
11	-5	1011
12	-4	1100
13	-3	1101
14	-2	1110
15	-1	1111

4ビットの場合

負の数の表現（2の補数）

+5　(2進表現　0 1 0 1)から　-5 の2進表現を得る

0 1 0 1 のビット反転
1 0 1 0 を得る。

それに 1 を加える

```
  1 0 1 0
+) 0 0 0 1
-----------
  1 0 1 1   ←  -5 の2進表現
```

4ビット
の場合

2の補数表現から10進数へ

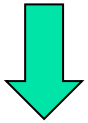
符号無　ud = b_3 x2x2x2 + b_2 x2x2 + b_1 x2 + b_0

符号付　sd = − b_3 x2x2x2 + b_2 x2x2 + b_1 x2 + b_0

Example:　2進表現　1 0 1 1

符号無　11

符号付　-5



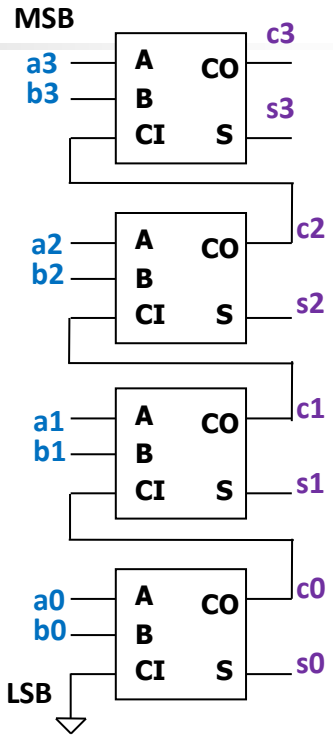
ud = 8 + 2+ 1 = 11

sd = -8 + 2 + 1 = -5

なぜ2の補数表現を用いるのか

2進演算	符号無	符号付
0 0 1 1	3	3
＋ 1 0 1 0	＋ 10	＋ -6
1 1 0 1	＝ 13	＝ -3

2の補数表現をすれば
符号付、符号無で
同じ2進演算アルゴリズム
同じハードウェアを
使用可能



(a3 a2 a1 a0) =(0 0 1 1)
(b3 b2 b1 b0)=(1 0 1 0)

(s3 s2 s1 s0) =(1 1 0 1)
(c3 c2 c1 c0) =(0 0 1 0)

負の数の表現（2の補数）

4,5ビットの場合

ビット数の拡張

4 ビット

符号無 ud	符号付 sd	2進 b3 b2 b1 b0
8	-8	1000
9	-7	1001
10	-6	1010
11	-5	1011
12	-4	1100
13	-3	1101
14	-2	1110
15	-1	1111

5 ビット

異なる !!

符号無 ud	b4 b3 b2 b1 b0	符号付 sd	b4 b3 b2 b1 b0
8	01000	-8	11000
9	01001	-7	11001
10	01010	-6	11010
11	01011	-5	11011
12	01100	-4	11100
13	01101	-3	11101
14	01110	-2	11110
15	01111	-1	11111

今から320年前、1692年のパリ

哲学者、数学者、科学者　ライプニッツ

（**Gottfried Wilhelm Leibniz**）

「全ての数を1と0によって表す驚くべき表記法」

を提案。

王立科学アカデミーに理解されず

学会誌にも掲載されなかった。

「誰も予想しなかった卓越した用途がありはずだ」

と語る。

（慶應義塾大学　青山友紀先生資料より）

ゴットフリート・ヴィルヘルム・ライプニッツ（Gottfried Wilhelm Leibniz, 1646年 - 1716年）

ライプニッツは哲学者、数学者、科学者など幅広い分野で活躍した学者・思想家として知られているが、また政治家であり、外交官でもあった。17世紀の様々な学問（法学、政治学、歴史学、神学、哲学、数学、経済学、自然哲学（物理学）、論理学等）を統一し、体系化しようとした。その業績は法典改革、モナド論、微積分法、微積分記号の考案、論理計算の創始、ベルリン科学アカデミーの創設等、多岐にわたる。ライプニッツは稀代の知的巨人といえる。

ブール代数 （2進数のための数学）

- 論理変数　A, B, C, …, Z
- 値は 1 または 0 をとる。（2値変数）

正論理　　1:　真 (true)
　　0:　偽 (false)
負論理　　0:　真 (true)
　　1:　偽 (false)

真にGND線を用いたいとき等に負論理を使用。

キーワード:

論理否定、論理積、論理和、排他的論理和、
真理値表、ド・モルガンの法則

ブール代数の創始者 George Boole (1815-1864、英)

コンピュータを理論的に支えるブール代数を提唱。

デジタル回路の設計には必須の知識。
デジタル回路は、電圧の High, Lowのみで
情報を演算するため、組み合わせ回路は
ブール代数での論理式で書き表わせる。

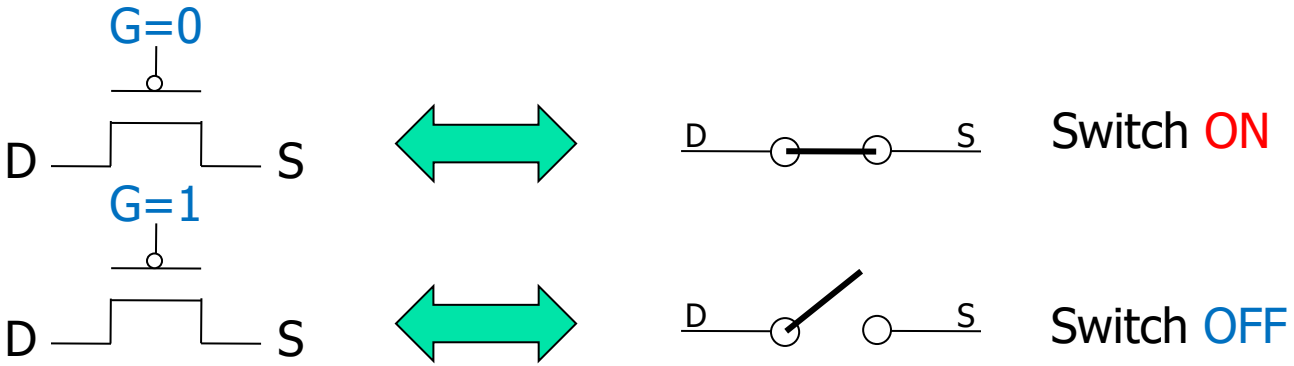
「記号の操作は計算の明白な要素として取り扱え
数量から切り離すことができる」

内　容

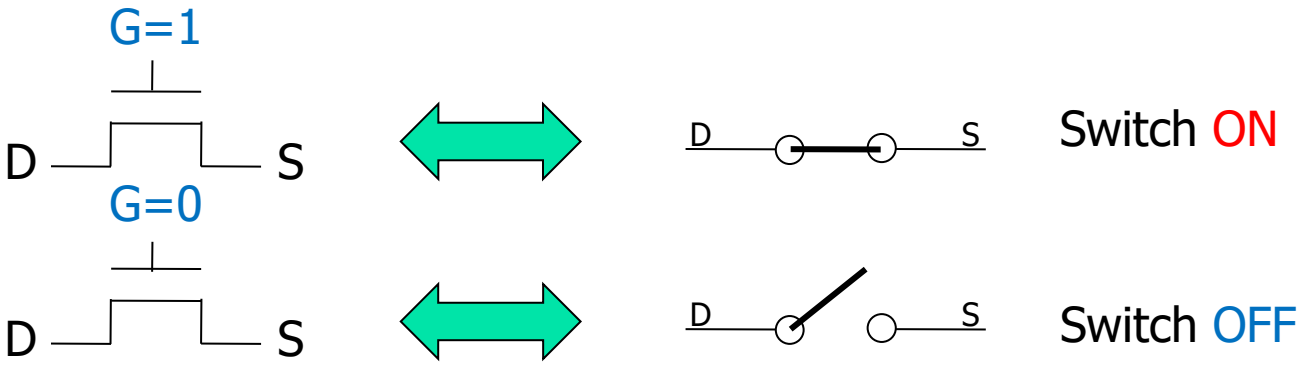
- セミナーの目的・目標
- デジタル回路の基礎
- スイッチレベル　デジタルCMOS回路
- デジタルCMOS回路の性能（消費電力、スピード）
- 同期回路設計とカウンタ回路
- 加算器、ビットシフト、乗算器
- 事例1：　SAR ADC+分散型積和演算回路
- 事例2：　自己校正機能をもったTDC回路
- 事例3：　冗長アルゴリズムを用いたSAR ADC
- 最後に

PMOS, NMOS スイッチ

(1) PMOS

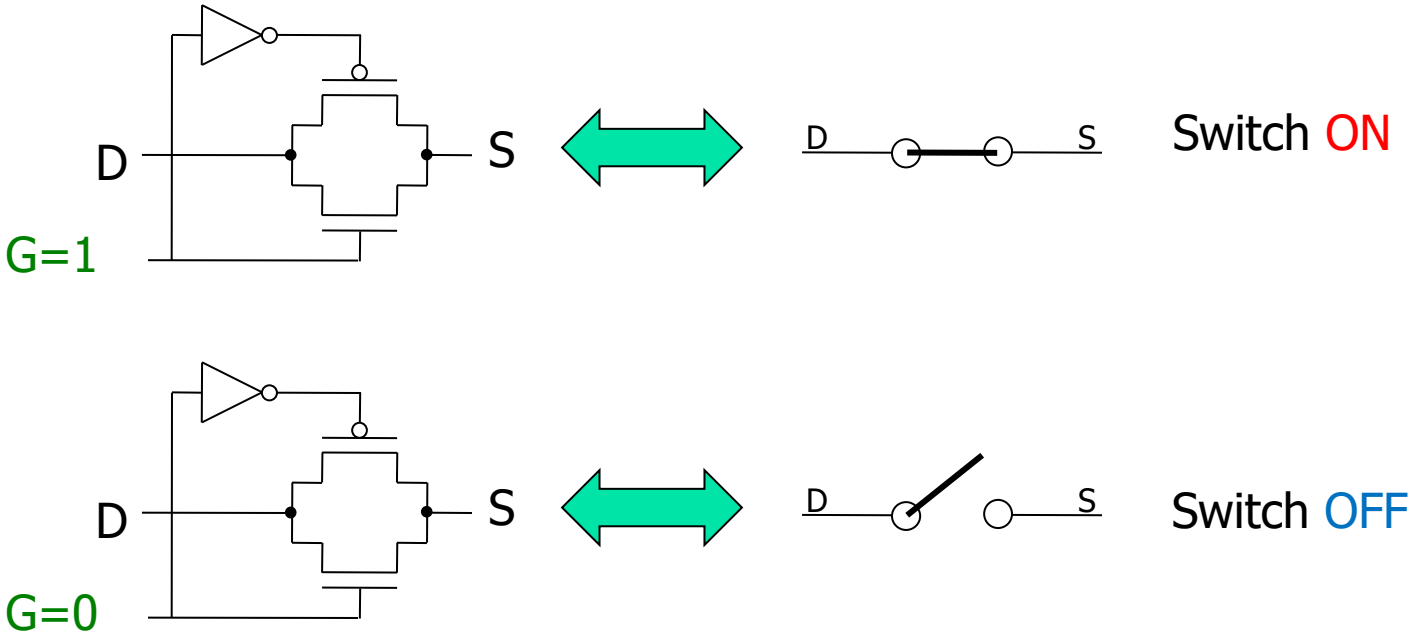


(2) NMOS



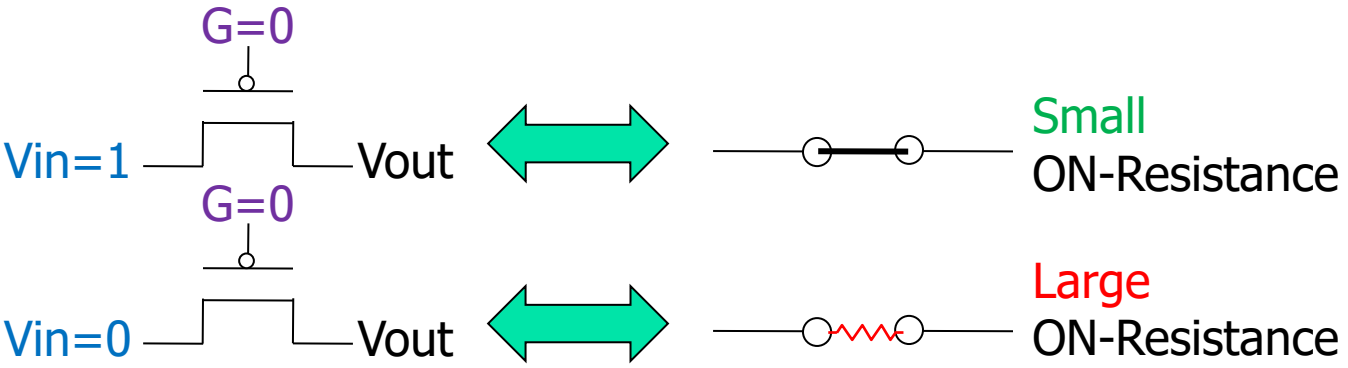
CMOSスイッチ

(3) CMOS



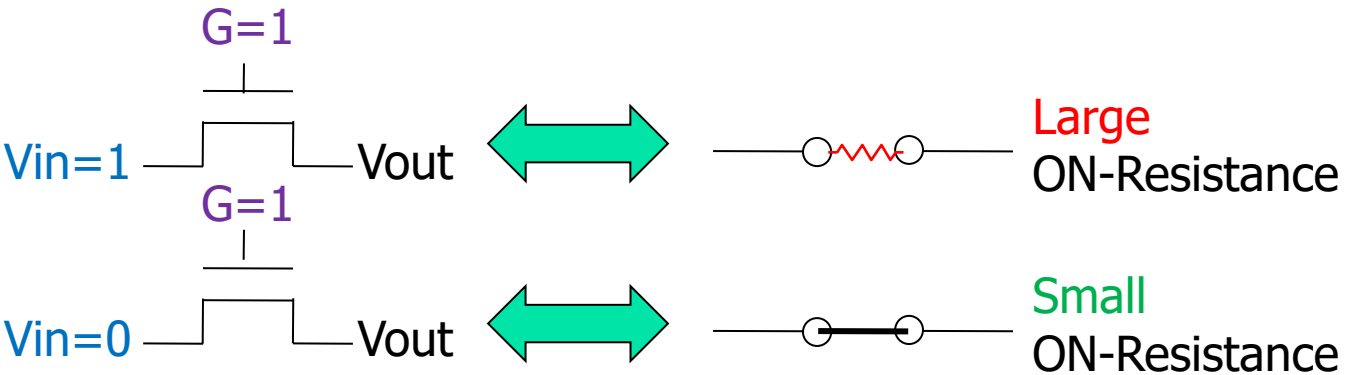
PMOS，NMOSスイッチのオン抵抗

(1) PMOS



PMOSは正電源側で用いる

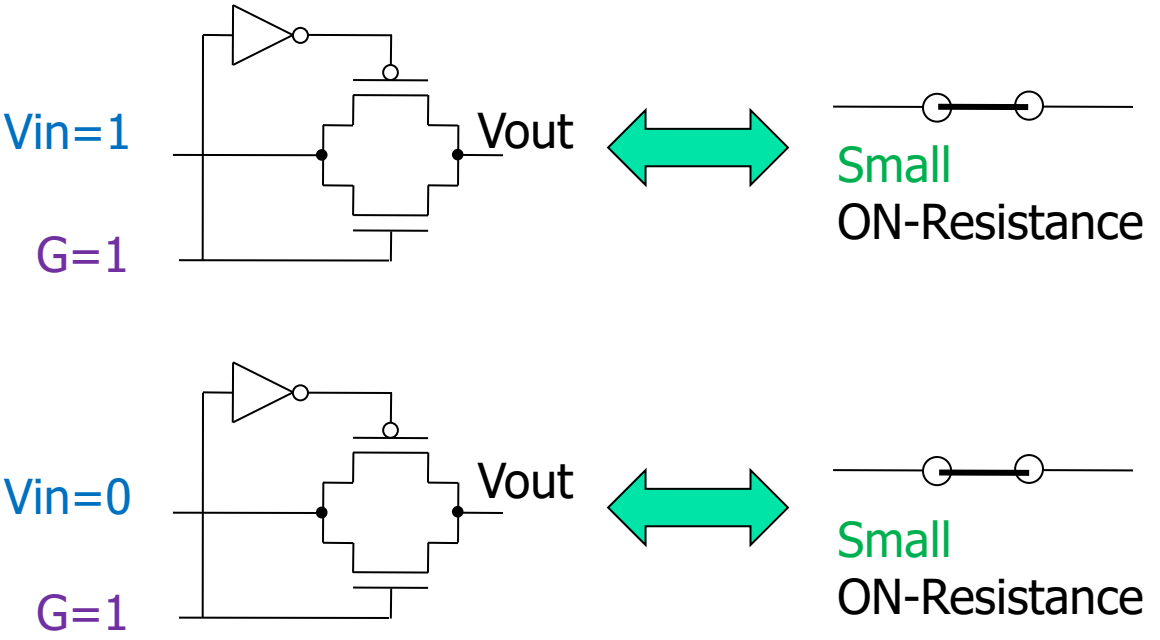
(2) NMOS



NMOSはGND側で用いる

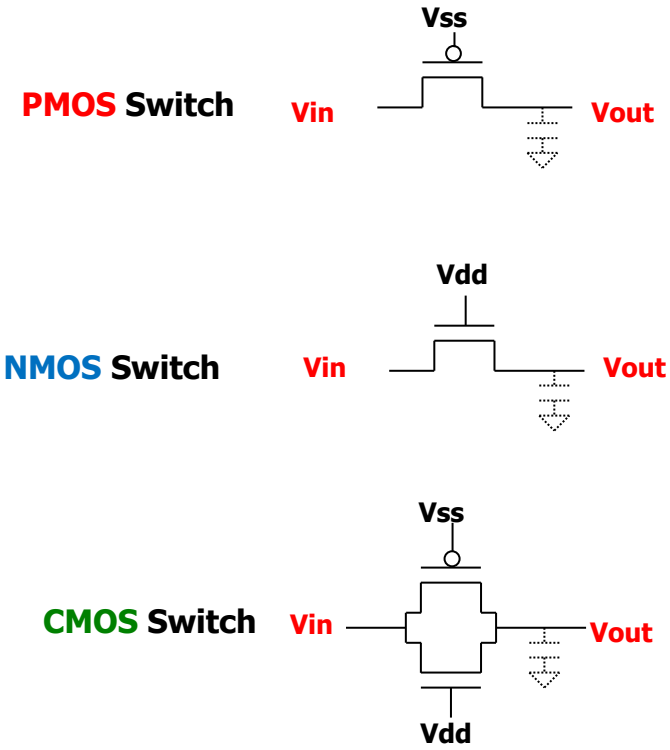
CMOSスイッチのオン抵抗

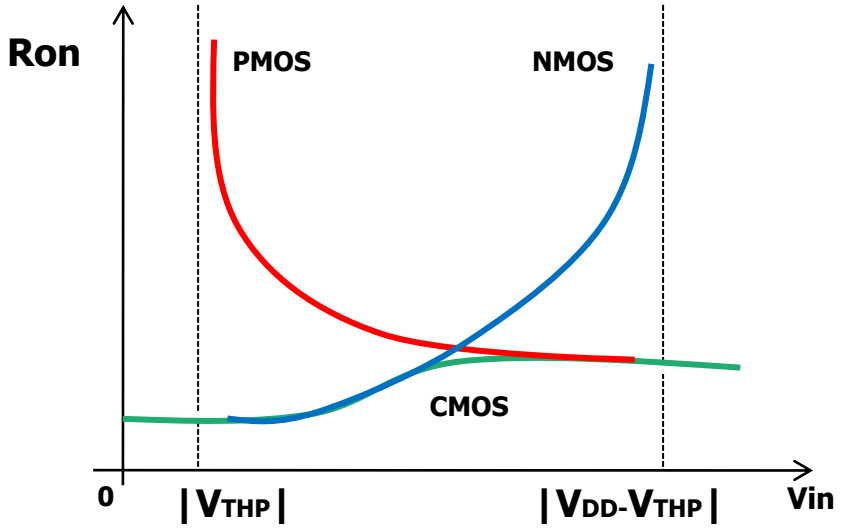
(3) CMOS



CMOSは
GND側でも
正電源側でも
オン抵抗が
小さいが、
トランジスタ数
が増える。

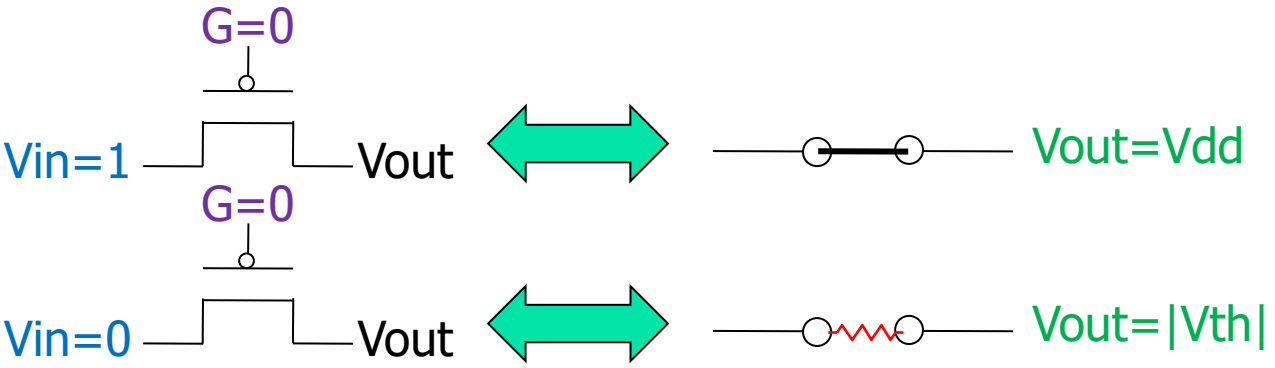
PMOS，NMOS，CMOSスイッチの入力電圧に対するオン抵抗





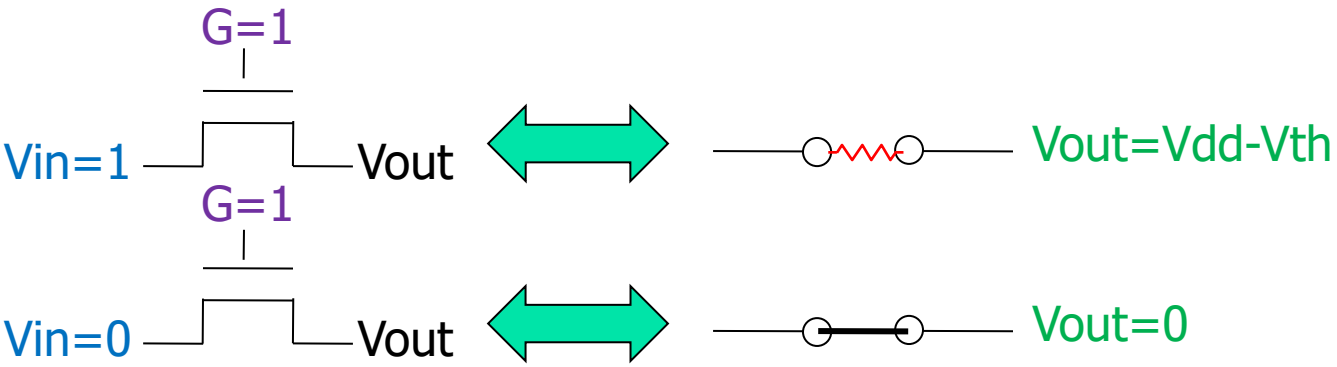
PMOS，NMOSスイッチの出力電圧

(1) PMOS



PMOSは
Vout は
GNDまで
下がらない。

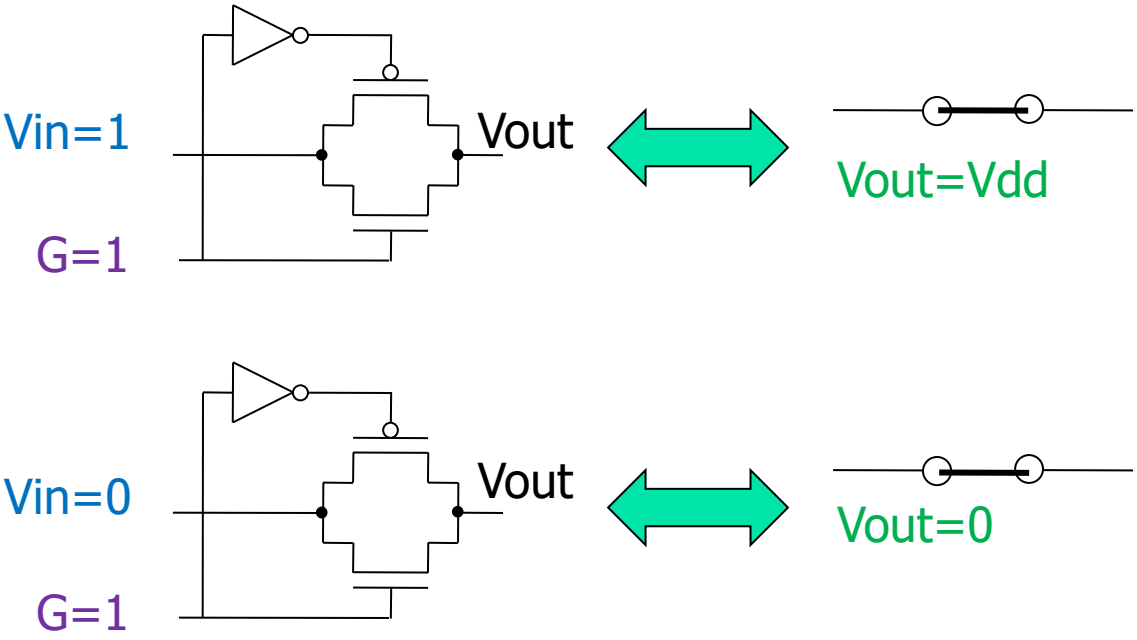
(2) NMOS



NMOSは
Voutが
Vdd まで
上がらない。

CMOSスイッチの出力電圧

(3) CMOS



CMOSでは
出力電圧Voutが
GND, Vdd 間を
フルスイング。

論理否定(NOT)

論理変数　A, Z

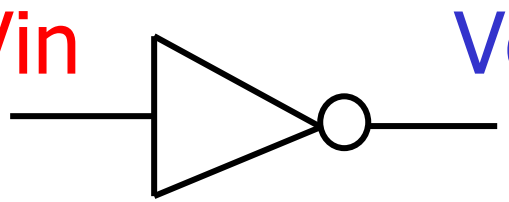
A:入力,　Z:出力

$Z = \overline{A}$

真理値表

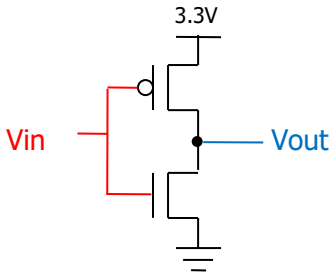
A	Z
0	1
1	0

NOT を実現する回路 ⇒ インバータ回路

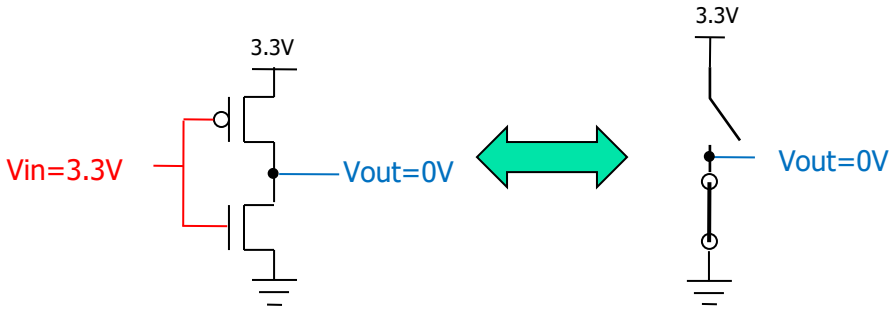


CMOSインバータ回路

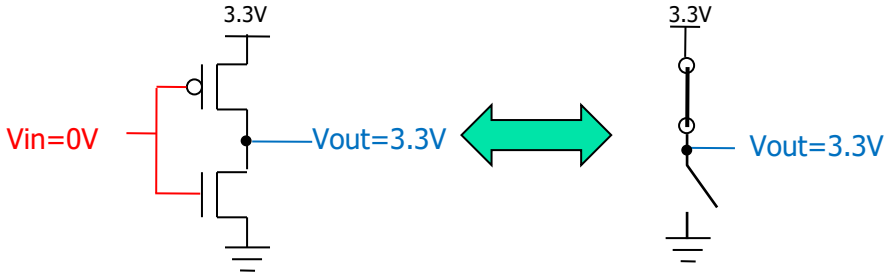
Inverter



a) When Vin=1 (3.3V)



b) When Vin=0



NAND　(NAND = AND + NOT)

論理変数　A,B, Z

A,B: 入力,　Z: 出力

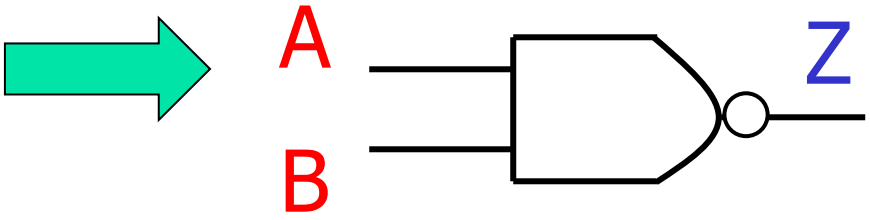
$Z = \overline{A \cdot B}$

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

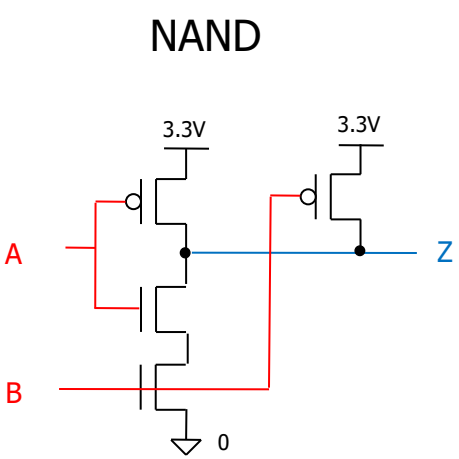
真理値表

NANDを実現する回路

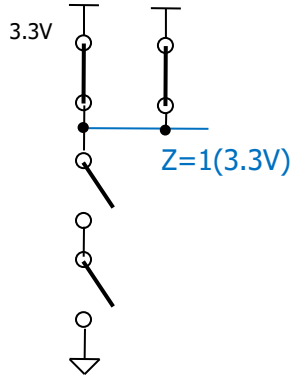
NAND回路



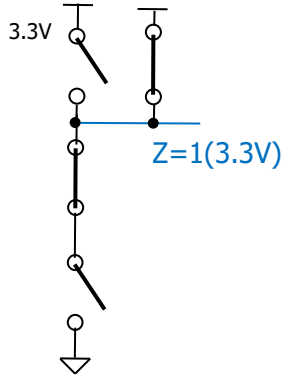
CMOS NAND回路



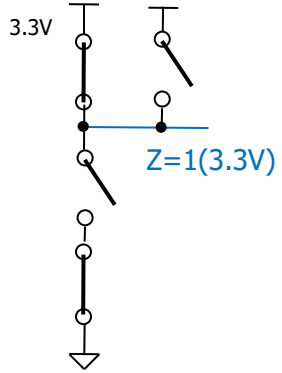
a) When A=0,B=0



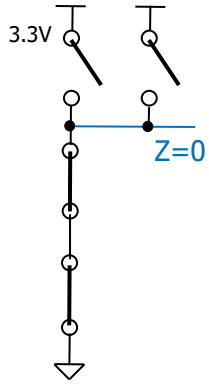
b) When A=1,B=0



c) When A=0,B=1



d) When A=1,B=1



NOR　（NOR = OR + NOT）

論理変数　A,B, Z

A,B: 入力,　Z: 出力

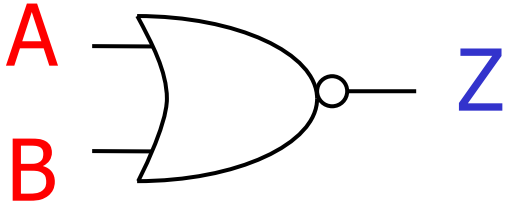
$Z = \overline{A+B}$

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

真理値表

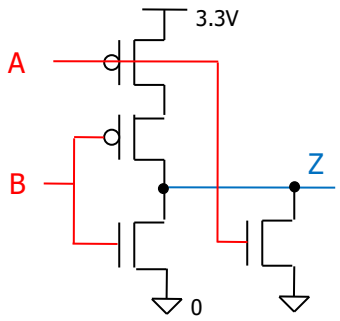
NORを実現する回路

NOR回路

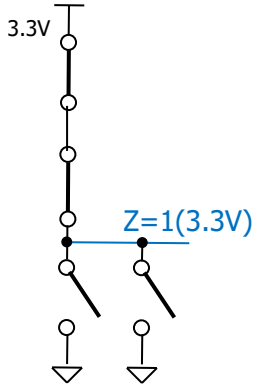


CMOS NOR回路

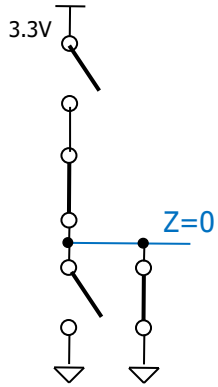
NOR回路



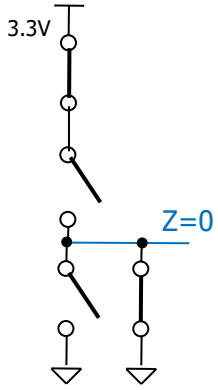
a) When A=0,B=0



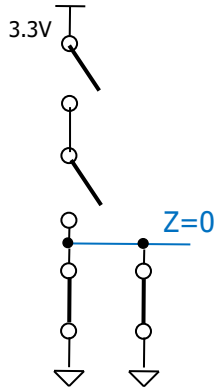
b) When A=1,B=0



c) When A=0,B=1



d) When A=1,B=1



論理和と2進数の加算

論理和　Z=　A＋B

右の真理値表は

論理和の定義

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

真理値表

論理和　　1+1=1

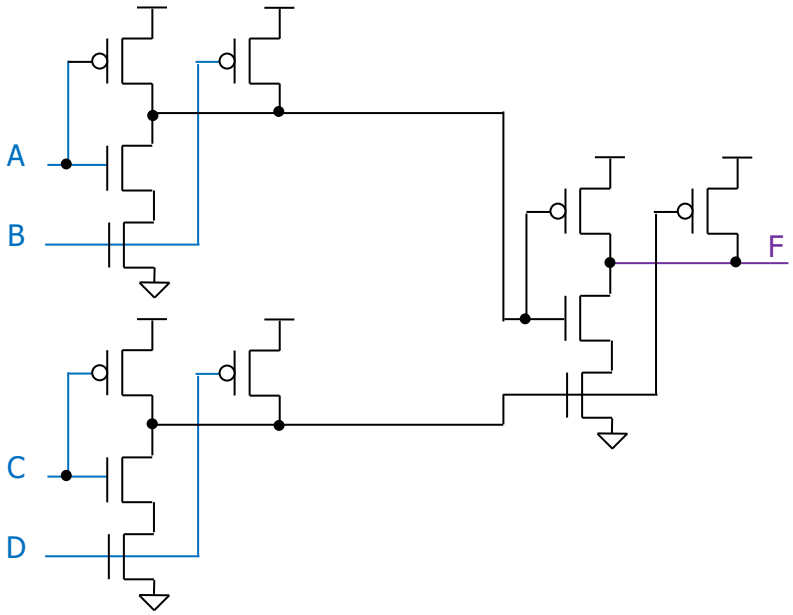
2進数の加算　1+1=10

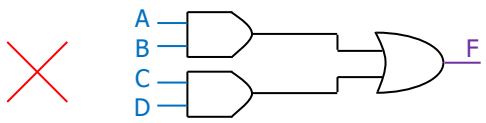
同じ 記号 ＋ でも意味が異なることに注意。

NAND、NOR回路を使用する。AND，OR回路の使用は避ける。

AND = NAND + Inverter
OR = NOR + Inverter
で構成 （非効率）

例：
F= A B + C D の回路実現

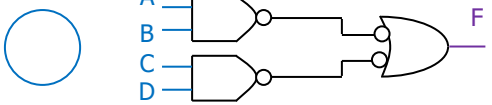


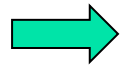


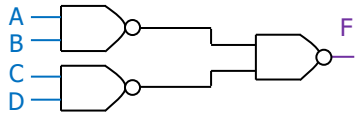
負論理

ド・モルガンの法則

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





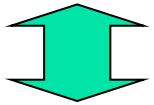


オーガスタス･ド･モルガン
Augustus De Morgan, 1806-1817

インド生まれのイギリスの数学者
ド･モルガンの法則を発案した。

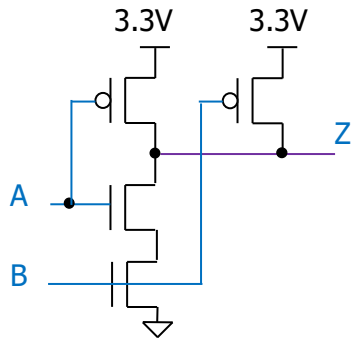
$$\overline{A \cdot B} = \overline{A} + \overline{B}, \quad \overline{A+B} = \overline{A} \cdot \overline{B}$$

「私の身長は 160 cm 以上であり、
かつ私の体重は 50 kg 以上である」ではない。



「私の身長は 160 cm 未満であるか、
または私の体重は 50 kg 未満である」

2入力NAND回路の
立上り、立下り時間はほぼ等しい
NAND回路は　NOR 回路より良い



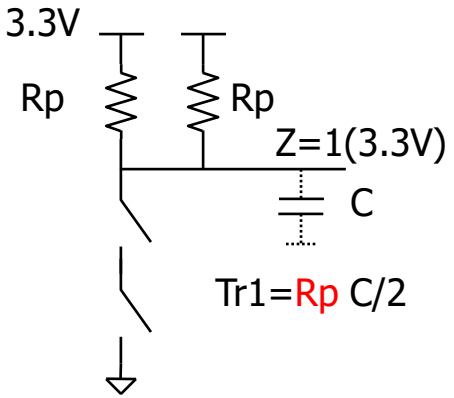
立上り時定数 Tr = Rp C
立下り時定数 Tf = 2 Rn C

Rp = 2～3 Rn

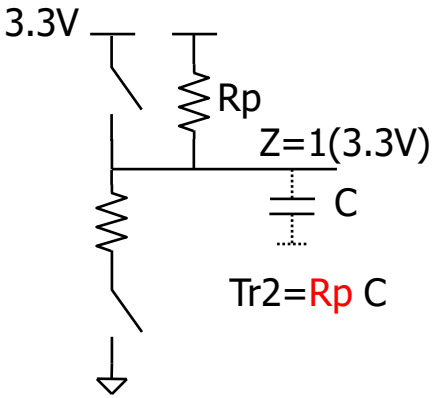
⇩

Tr = Tf

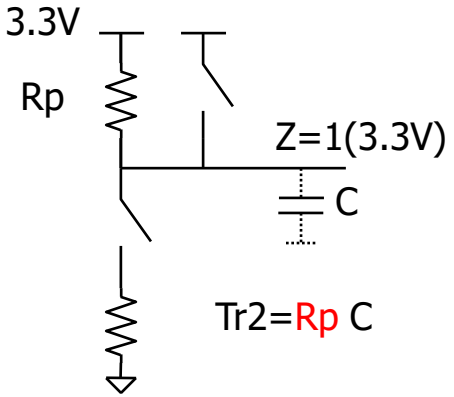
a) When A=0,B=0



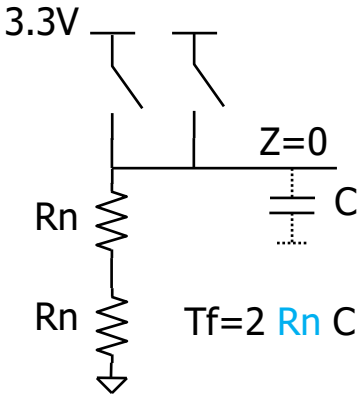
b) When A=1,B=0



C) When A=0,B=1



D) When A=1,B=1

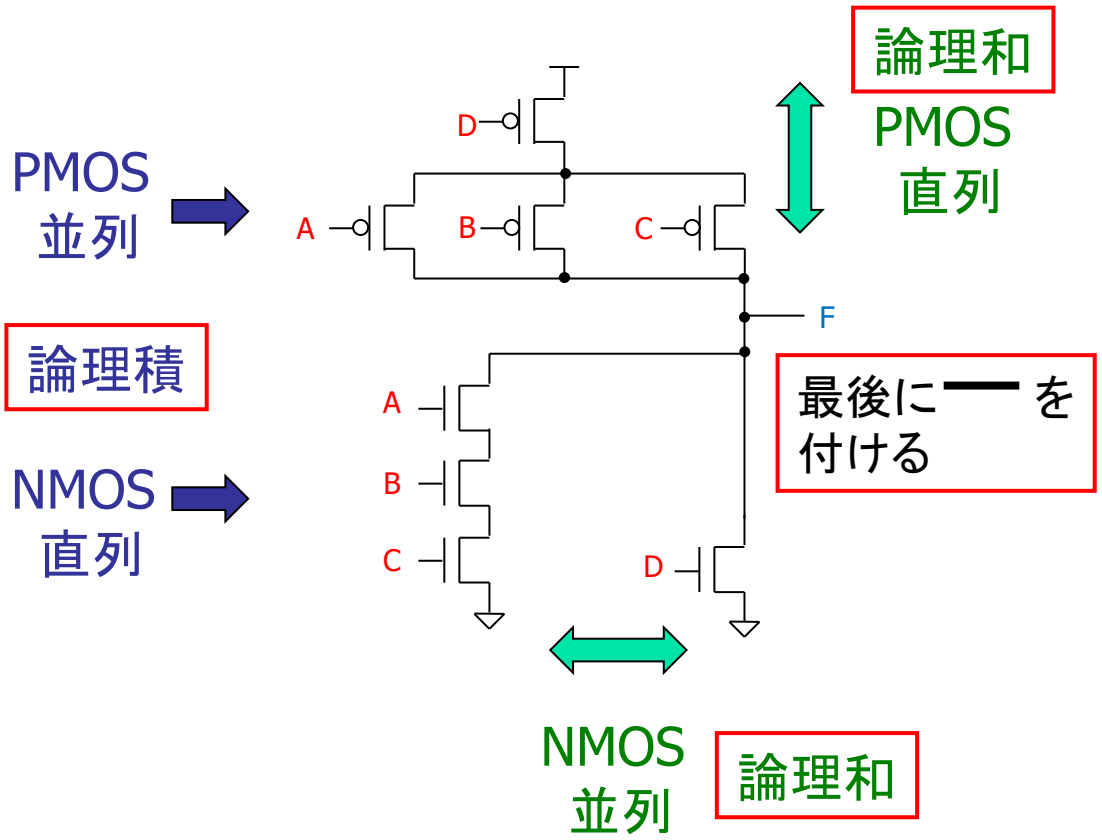


複合論理CMOS回路

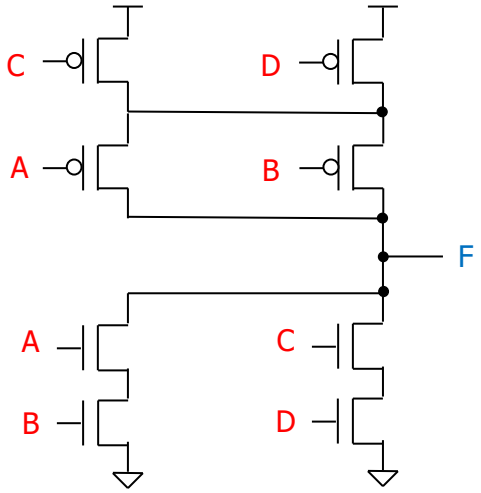
	論理積	論理和
PMOS	並列	直列
NMOS	直列	並列

複合論理CMOS回路　例

$F = \overline{A \cdot B \cdot C + D}$



$F = \overline{A\,B + C\,D}$



マルチプレクサ

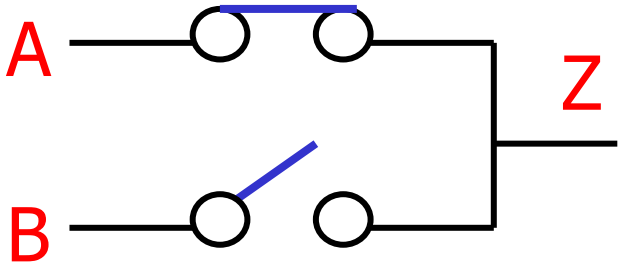
論理変数　A,B,S,Z

A,B,S:入力,　Z:出力

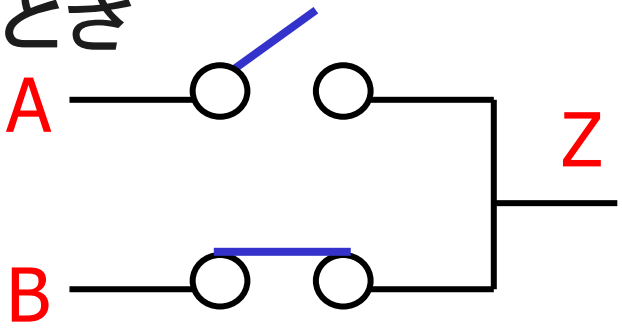
S	Z
0	A
1	B

真理値表

S=0 のとき

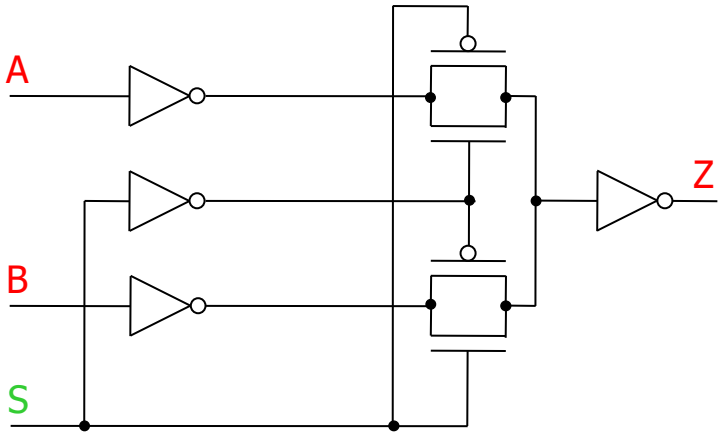


S=1 のとき

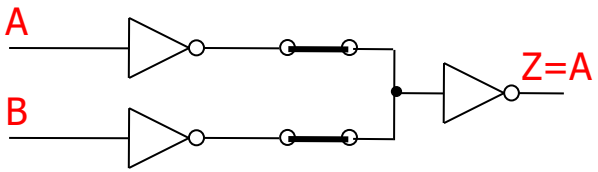


CMOS マルチプレクサ回路

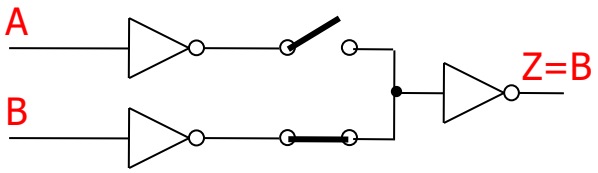
Multiplexer



a) When S=0



b) When S=1



排他的論理和(EXOR)

論理変数　A,B, Z

A,B: 入力,　Z: 出力

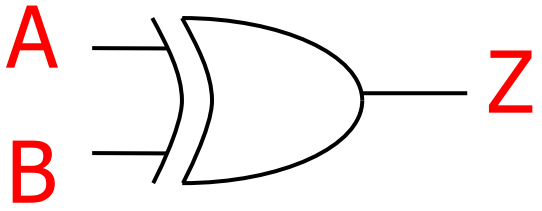
Z=　A⊕B

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

真理値表

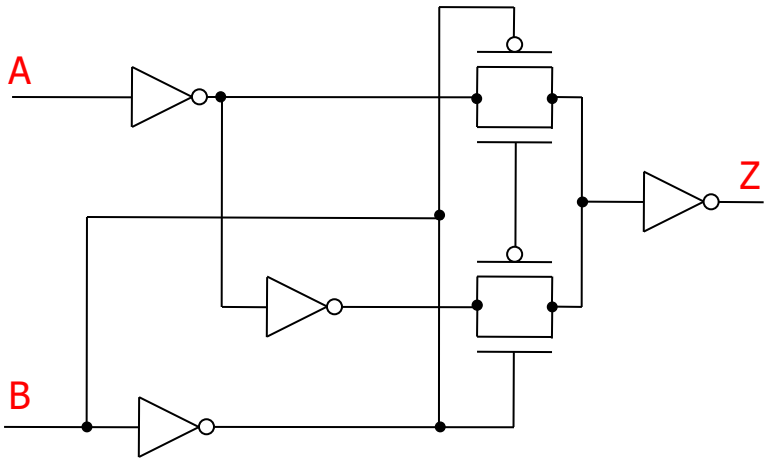
EXORを実現する回路

EXOR回路

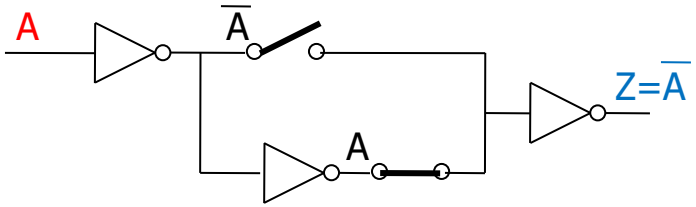


CMOS EXNOR回路

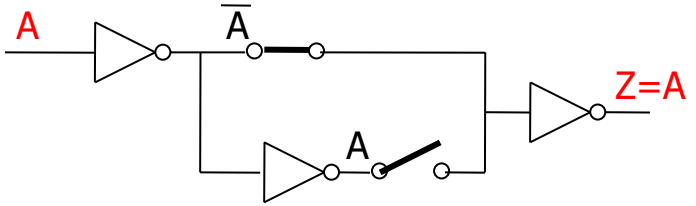
$Z = A\,B + \overline{A}\,\overline{B}$



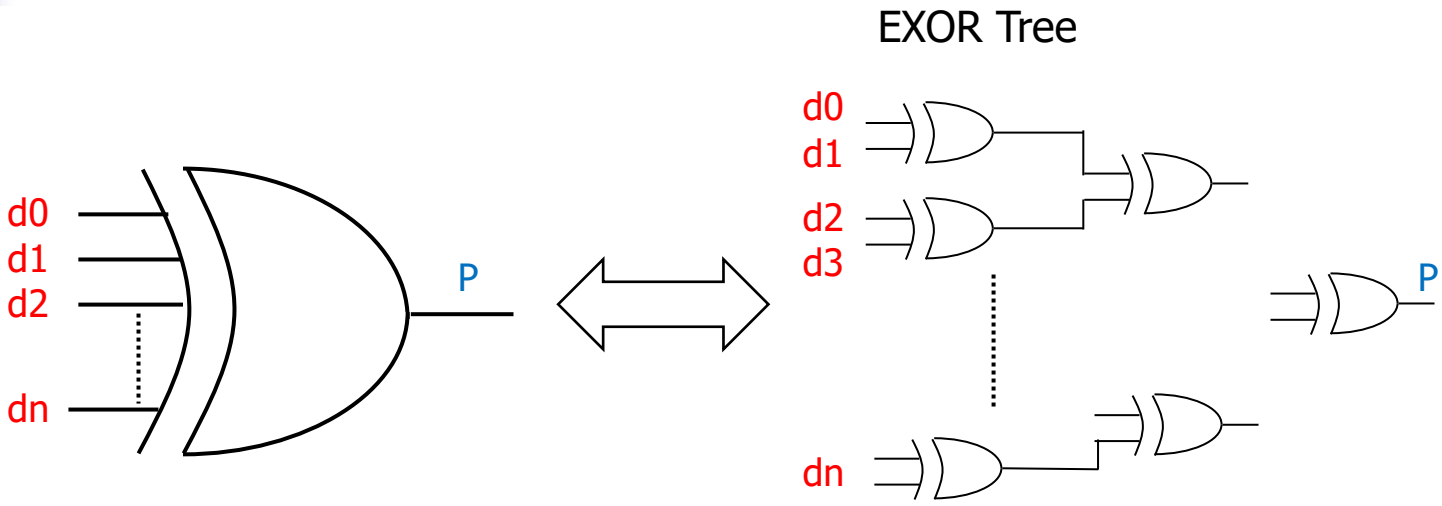
a) When B =0



b) When B =1



多入力EXOR 回路とパリティ



$P = d0 \oplus d1 \oplus d2 \oplus \cdots\cdots \oplus dn$

Parity

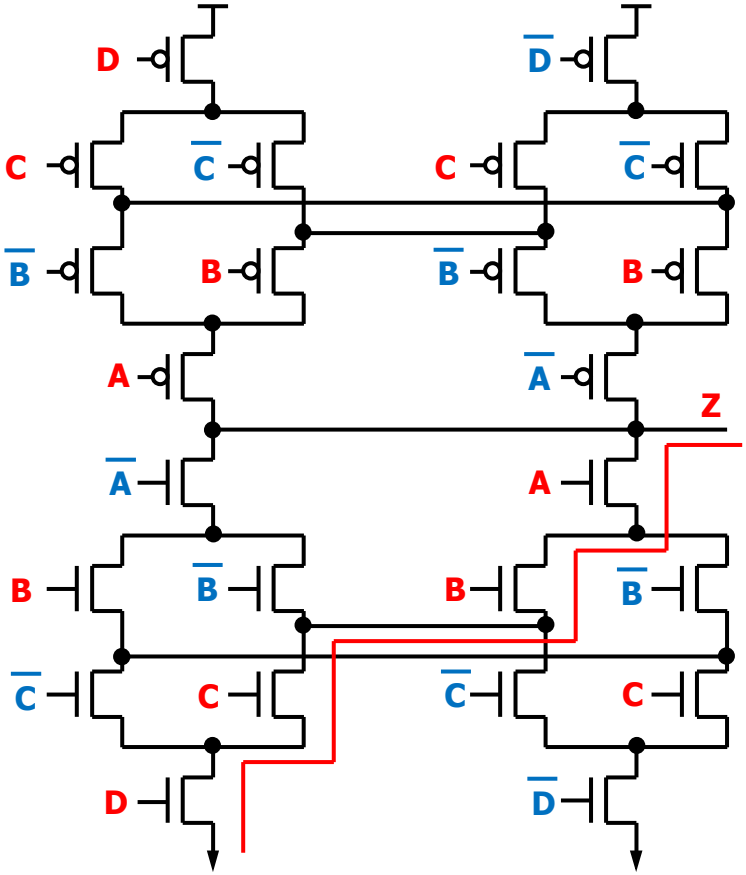
d0, d1, … dn の1の数が奇数個 ⟹ P=1

偶数個 ⟹ P=0

メモリ、通信のデータチェック等に使用

４入力EXORの実現回路

$Z = A \oplus B \oplus C \oplus D$



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

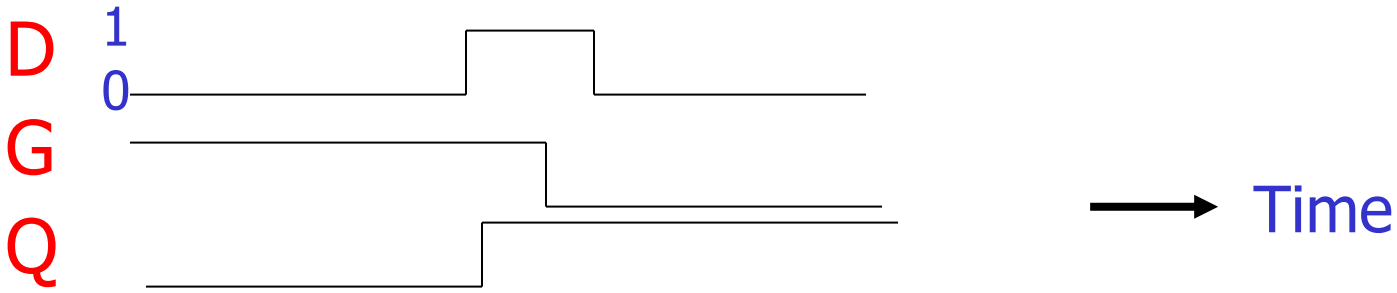
情報記憶素子(ラッチ)

論理変数　D, G, Q

　D, G: 入力,　Q: 出力

G=1 のとき　　Q=D

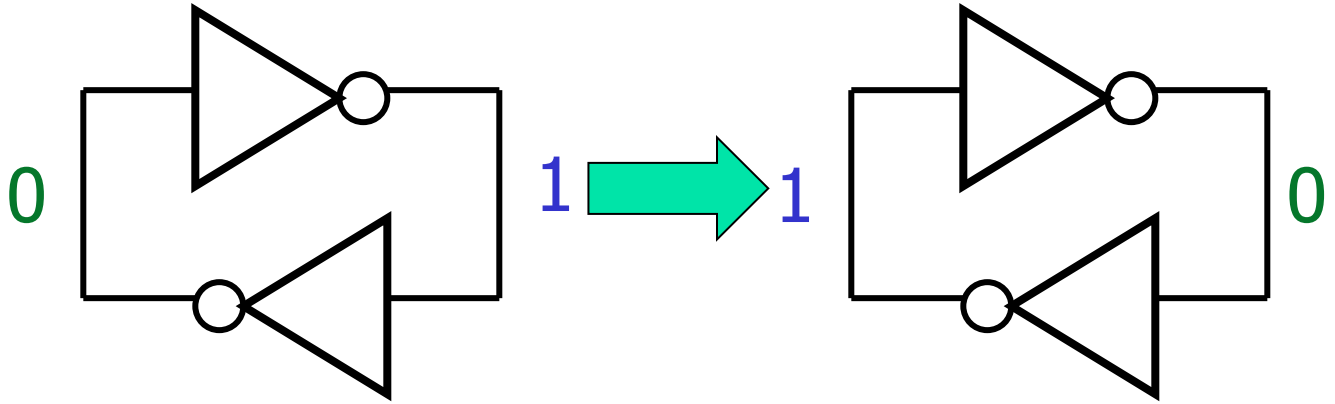
G=0 のとき　　Qは Gが1からOになる瞬間の
　　Dの値(1 or 0) を保持(記憶)している。



2つのインバータのリング接続 メモリ回路

2つの安定状態

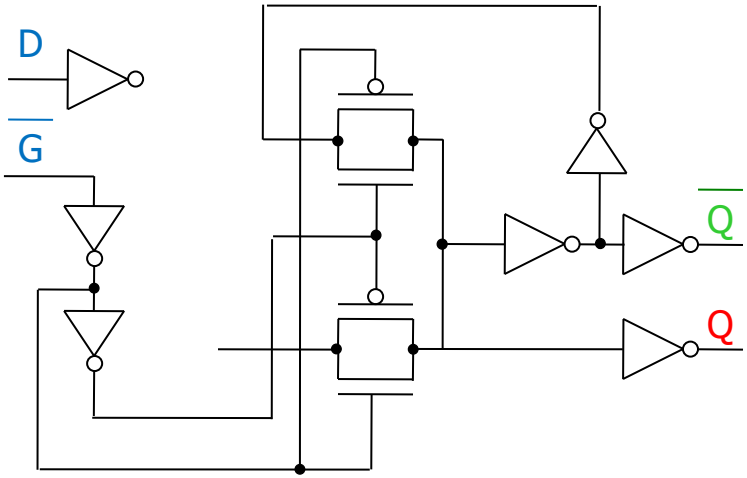
データ“1”を記憶　　データ“0”を記憶



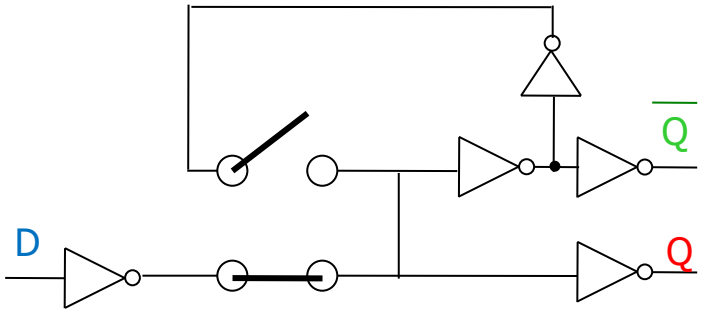
- SRAM (Static ランダム・アクセス・メモリ)
  Latch, Flip-Flop 等のメモリ素子は
  これを利用している。

CMOS ラッチ回路

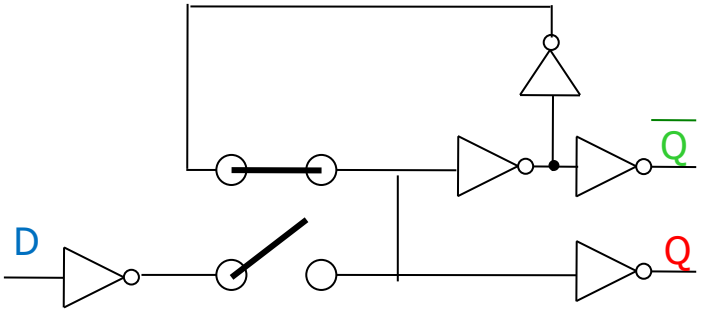
Latch回路
（メモリ素子）



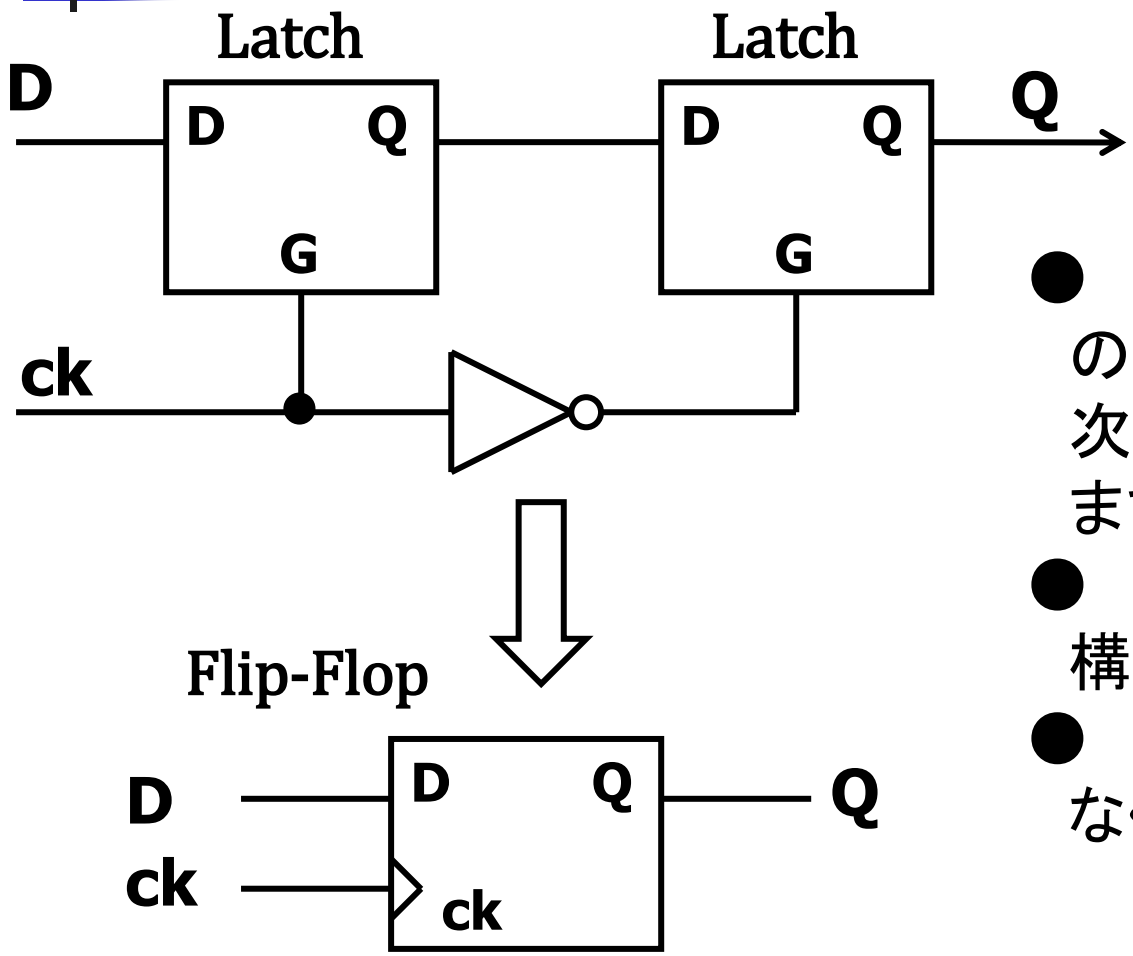
a) When $\overline{G}$ =0　トラック・モード



b)When $\overline{G}$ =1　ラッチ・モード

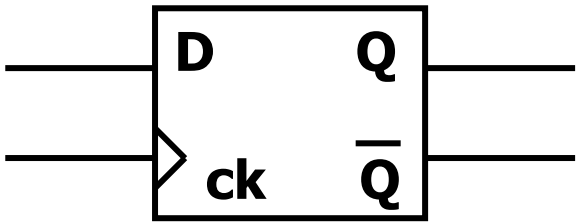


フリップ・フロップ回路

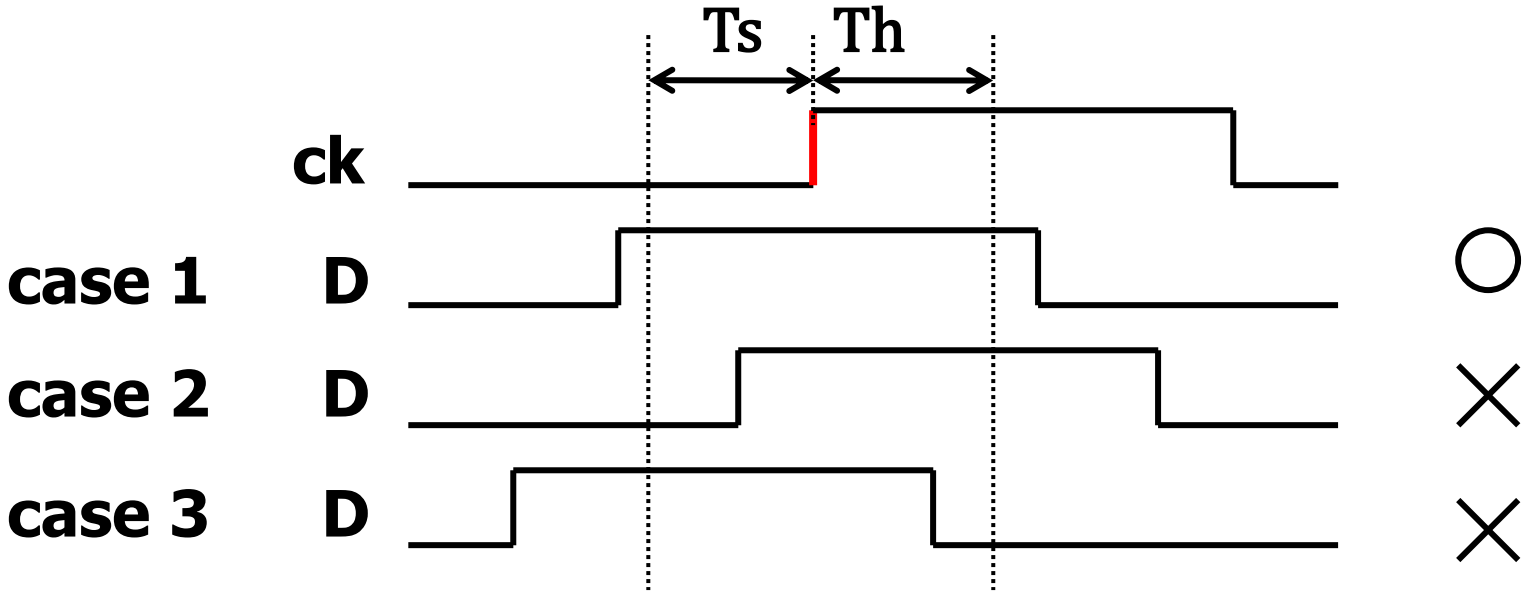


- クロックck の立ち上がりの瞬間のデータDを次のクロック立ち上がりまで（CK＝1，0でも）保持
- 2つのラッチ回路から構成
- 高速回路ではラッチではなくフリップフロップを使用

フリップ・フロップ回路と セットアップ時間、ホールド時間



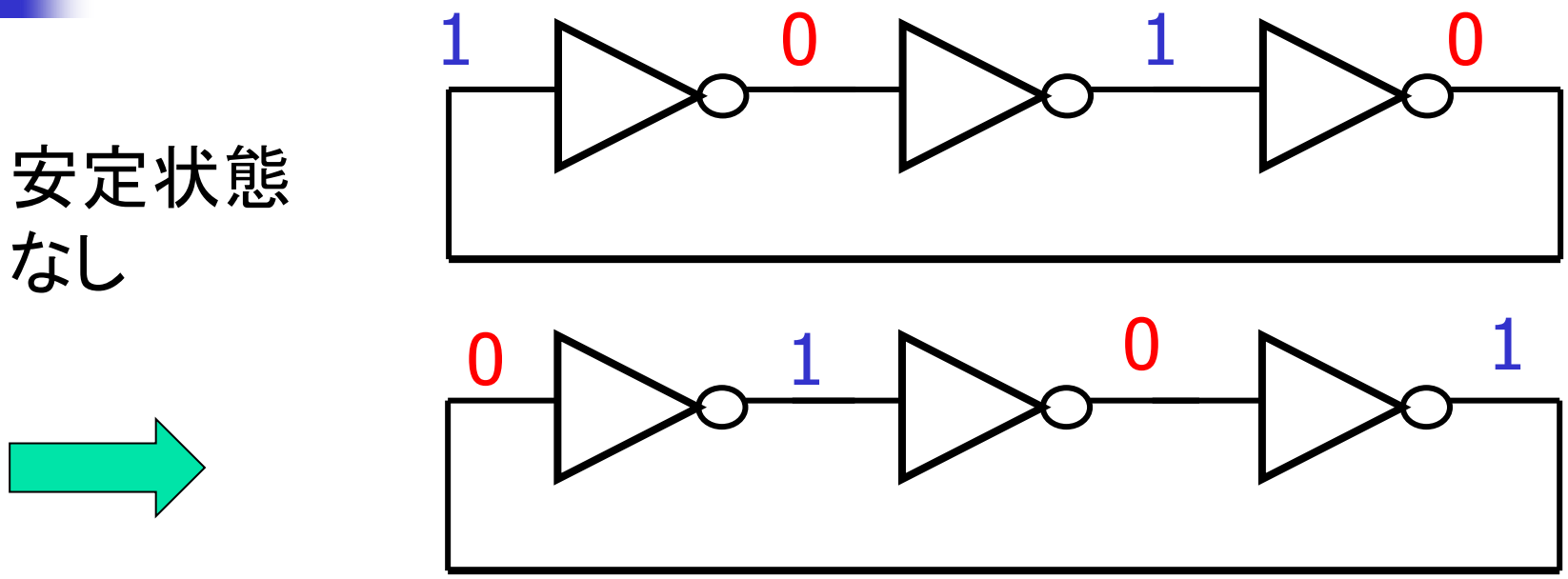
Ts : set-up time
Th : hold time



Set-up Time & Hold Time of Flip-Flop

奇数個インバータのリング接続
リング発振器

安定状態
なし



T: インバータ遅延、 2N+1 個のインバータリング接続

周波数 $f = \dfrac{1}{2\,(2N+1)\,T}$ で発振する。

内　容

- セミナーの目的・目標
- デジタル回路の基礎
- スイッチレベル　デジタルCMOS回路
- **デジタルCMOS回路の性能（消費電力、スピード）**
- 同期回路設計とカウンタ回路
- 加算器、ビットシフト、乗算器
- 事例1：　SAR ADC+分散型積和演算回路
- 事例2：　自己校正機能をもったTDC回路
- 事例3：　冗長アルゴリズムを用いたSAR ADC
- 最後に

エネルギーとパワー

- エネルギー　[Joule]

  電力(パワー)　[Watt]

  Joule = Watt・s

  電力は単位時間当たりに消費されるエネルギー

  電力＝電圧･電流　P = V･I

- 電流:　単位時間当たりに流れる電荷量

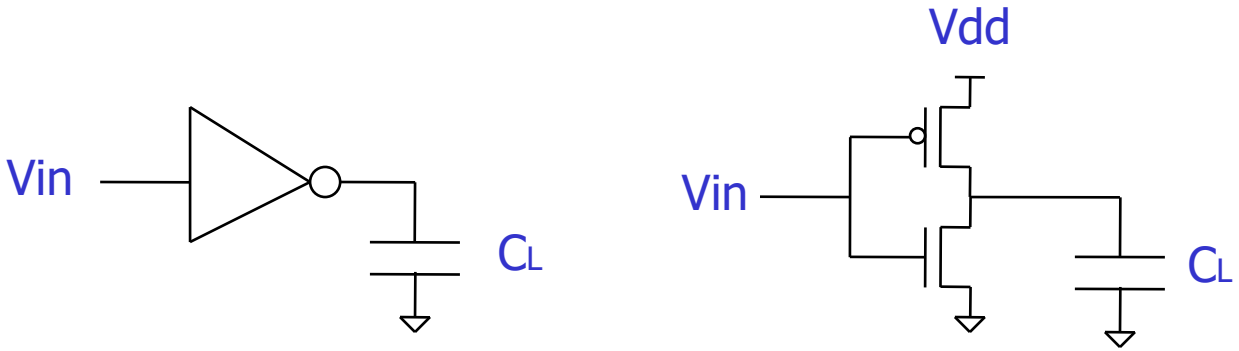
デジタルCMOS回路の電力消費

デジタルCMOS回路（インバータ）

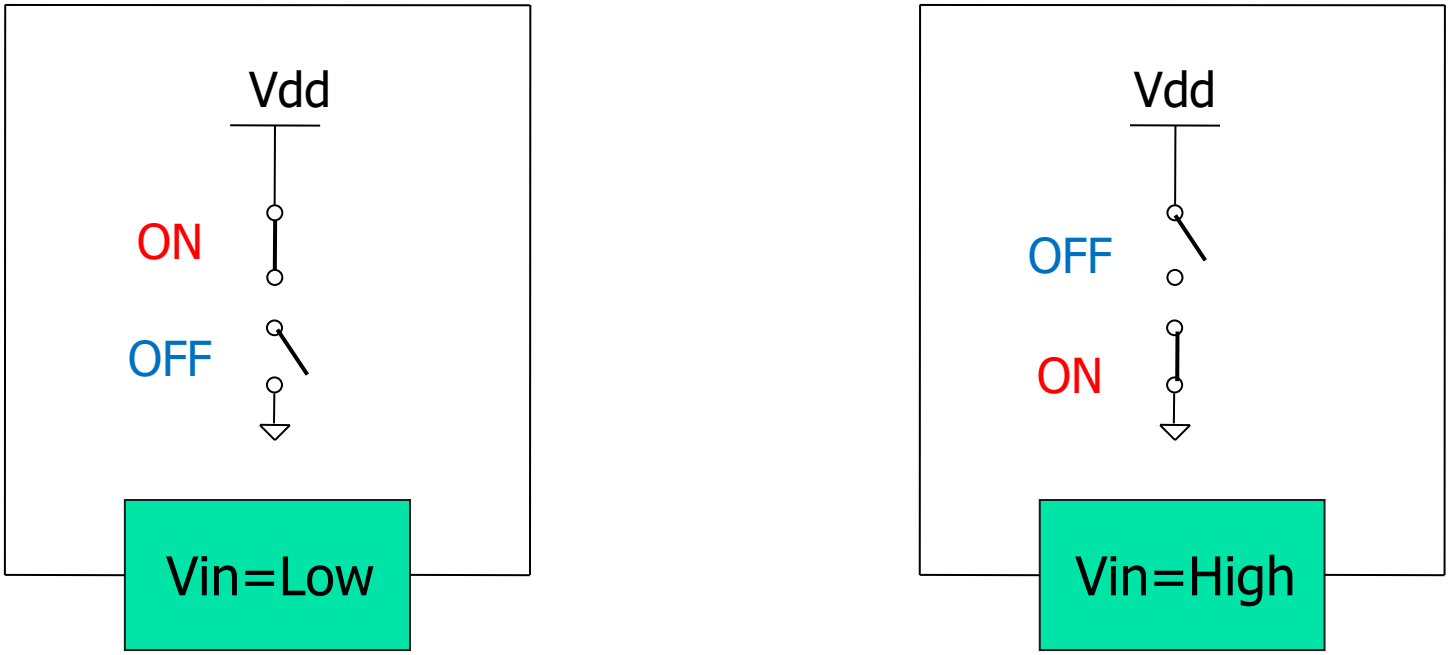
Vdd: 電源電圧

Vin: 入力、　Vout: 出力

C_L ： 負荷容量

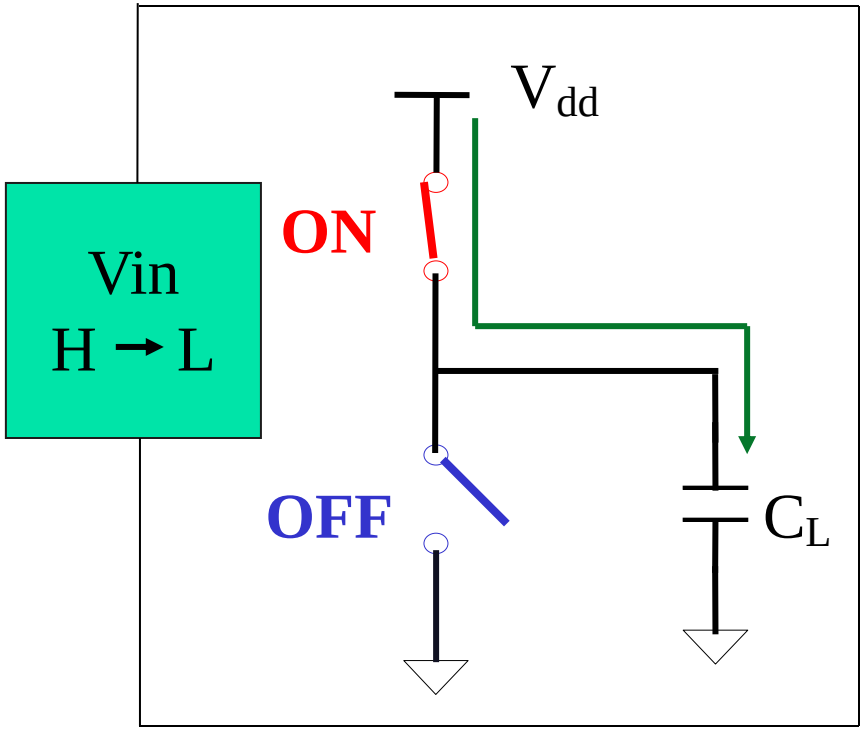


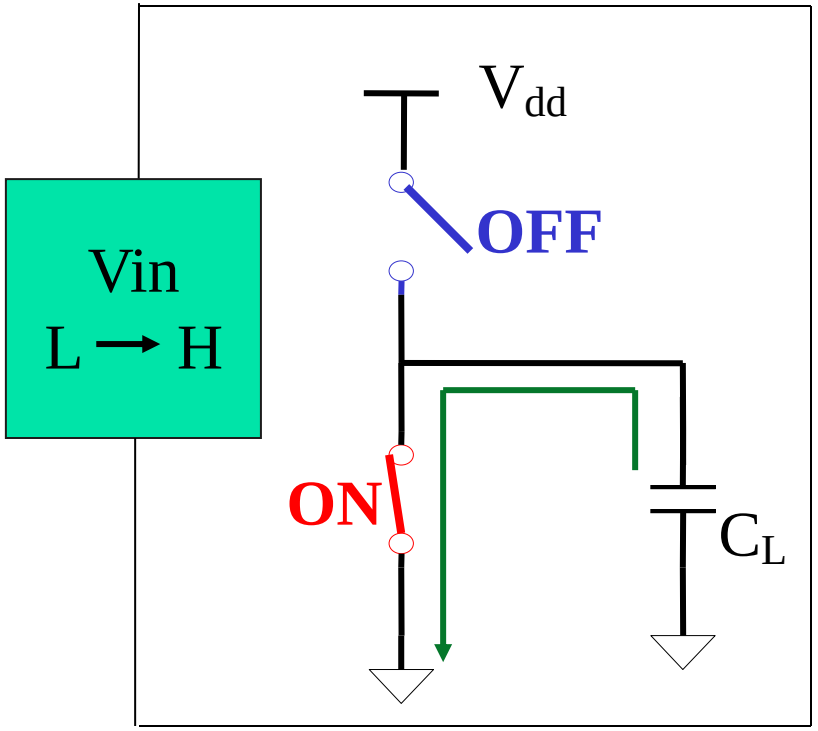
静的電力消費は小さい



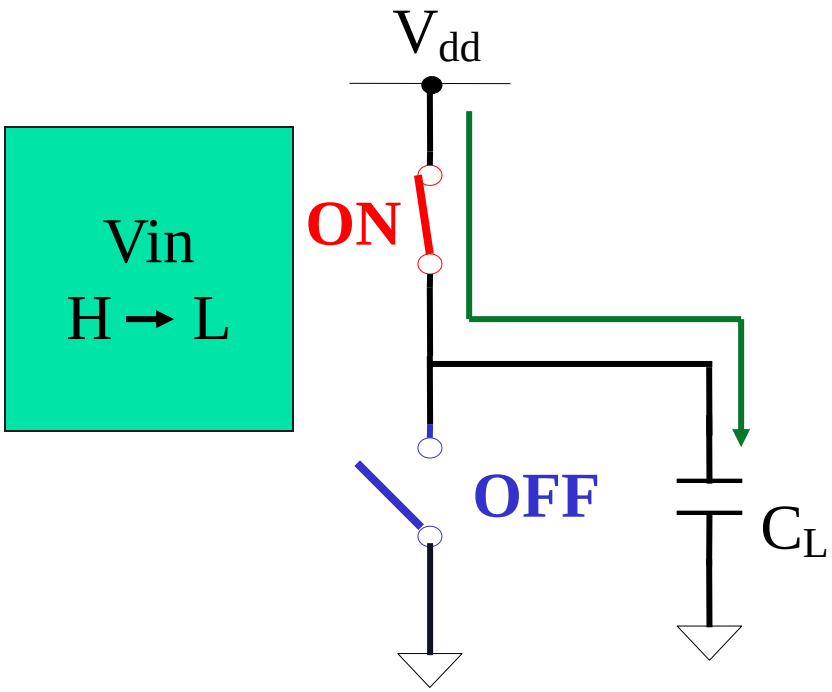
（注） 最近の微細CMOSデジタル回路では リーク電流が大きくなり、静的電力消費の占める割合が増えてきている。

動的消費電力 (1)





動的消費電力 (2)



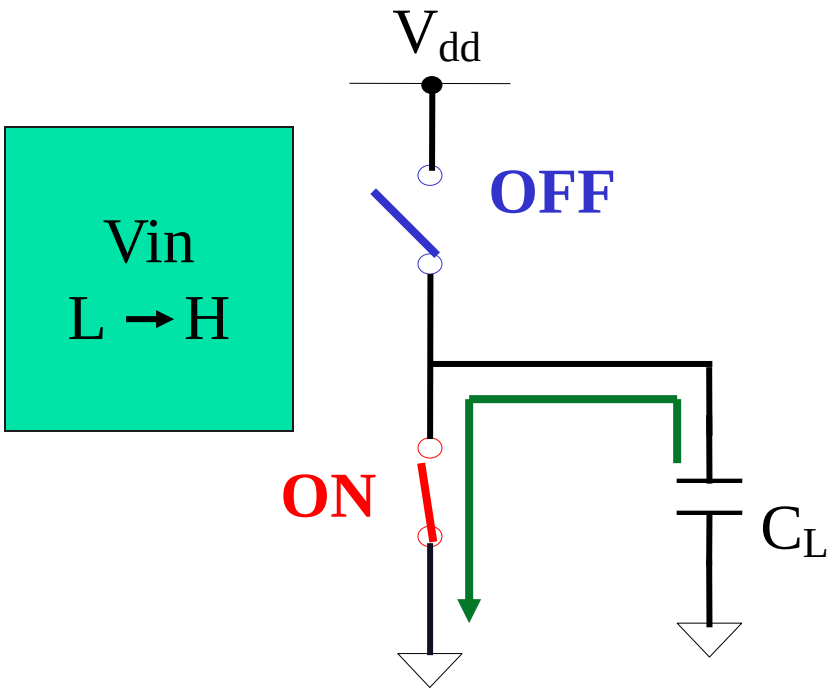
入力Vin

High → Low

蓄積電荷Q

0 → $C_L V_{dd}$

動的消費電力 (3)



入力Vin

Low → High

蓄積電荷Q

CL Vdd → 0

動的消費電力 (4)

V_{in} : H ⟶ L ⟶ H のとき

電荷 $Q = C_L V_{dd}$ が電源 V_{dd}から GND へ流れる。

一秒間に出力が f 回のトグルするとき

Vdd からGNDへ流れるトータルの電荷 $Q_{total} = f\, C_L\, V_{dd}$

$$\therefore \text{消費電力} \qquad P = V_{dd} \cdot I$$
$$= V_{dd}(f \cdot C_L \cdot V_{dd})$$
$$= f \cdot C_L \cdot V_{dd}^{\,2}$$

f : 出力トグル周波数　　C_L : 負荷容量

V_{dd} : 電源電圧

デジタルCMOS VLSIの低消費電力化

低消費電力化は大きな技術的課題

例： 携帯電話 ⇒ バッテリーが長持ちさせる

低消費電力化技術 ⇒ f, CL, Vdd を小さくする。

技術のトレンド：

周波数 f ：マイクロプロセッサのクロック周波数はより高くなる。

X

寄生容量CL： 半導体の微細化により寄生容量は小さくなりつつある。 ○

電源電圧Vdd: より低くして用いる。

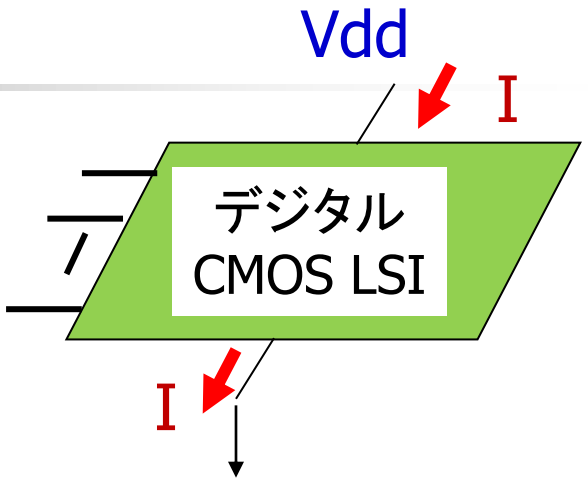
5V ⇒ 3.3V ⇒ 1.8V ⇒ 1V ○

デジタルCMOS 回路の低消費電力化技術　例

低消費電力化技術 ⇒ f, C_L, Vdd を小さくする。

- 0, 1 のトグル回数の多いノード (f の高いノード)の負荷容量C_L を小さくする。
  - ⇒ そのノードの配線長を短くする (配線容量を小)
    - レイアウトが低消費電力化に貢献
  - Fan-out を小さくする
- ノードの0, 1 のトグル回数の少ないアルゴリズムを開発する
  - ⇒ 数学が低消費電力化に貢献

デジタルCMOS LSI に対する
IDDQ テスト



各入力ピンを
Vdd または
GNDに固定

デジタルCMOS LSI

テスト時に入力を固定（トグルしない）

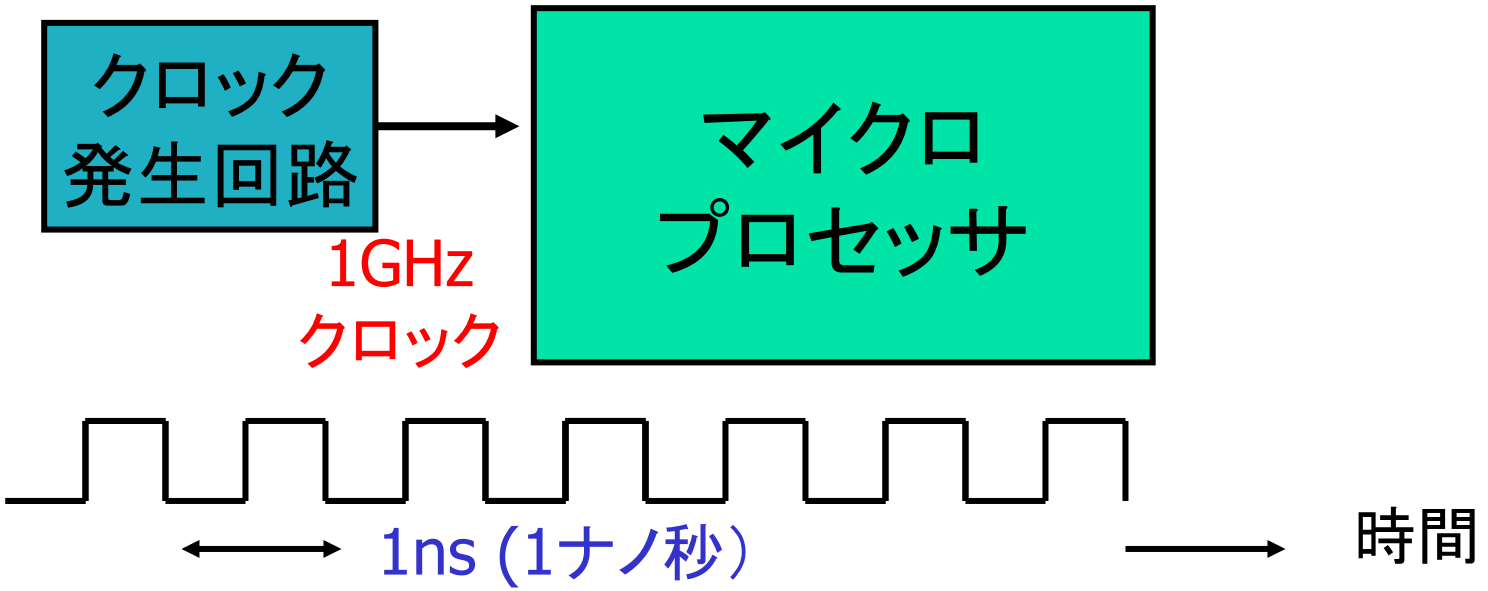
正常チップ：電源VddからGNDへの電流 I は微小

電流Iが大きければ「どこかに故障あり」と判定

http://mix.kumikomi.net/index.php/IDDQテスト

マイクロプロセッサのクロック

- クロックに同期して動作（同期回路）
  クロックの立ち上がりで論理回路はトグル。
- より高い周波数になってきている。



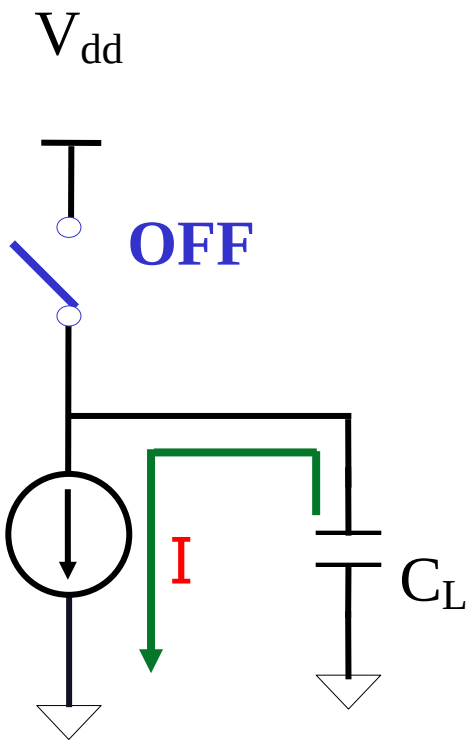
デジタルCMOS 回路のスピード

電源電圧 Vdd:

- 低消費電力化のため電源電圧を下げると
  スピードは遅くなる。
- スピードは電源電圧に比例
- 消費電力は電源電圧の２乗に比例

温度: スピードは温度にほぼ反比例。
電子、正孔の移動度が温度上昇とともに減少のため

なぜ電源電圧を上げると デジタルCMOS回路は高速化するのか？



引き抜く電荷
Q=C Vdd

MOSの２乗則

$I = K (Vdd-Vth)^2$

$\fallingdotseq K\ Vdd^2$

ゲート遅延

$T = Q / I$

$= C / (K\ Vdd)$

デジタル回路の Figure of Merit (FOM)

FOM = スピード/消費エネルギー

「A」のエネルギーを消費し「B」のスピードの回路と、
「2A」のエネルギーを消費し「2B」のスピードの回路の
FOM は同じ。

工学設計：トレードオフ（Trade-off, 妥協）
の考え方が重要

デジタルCMOS回路：
電源電圧を小さくして使用するとFOMが良。

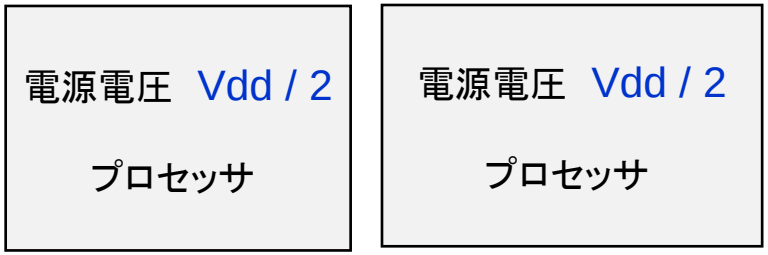
並列処理による低消費電力化

ケース1： 電源電圧 Vdd, 1つのプロセッサ

消費電力 $= K (Vdd)^2$

処理スピード $= L\ Vdd$

ケース2： 電源電圧 Vdd/2, 2つのプロセッサ



消費電力 $= K (Vdd/2)^2 + K (Vdd/2)^2$

$= (1/2)\ K (Vdd)^2$

処理スピード

$= (1/2)\ L\ Vdd + (1/2)\ L\ Vdd$

$= L\ Vdd$

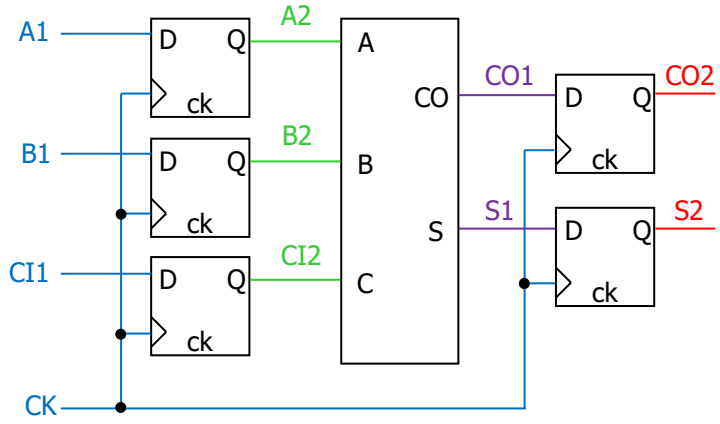
ケース2はケース1と処理スピードは同じであるが、消費電力は1/2 になる

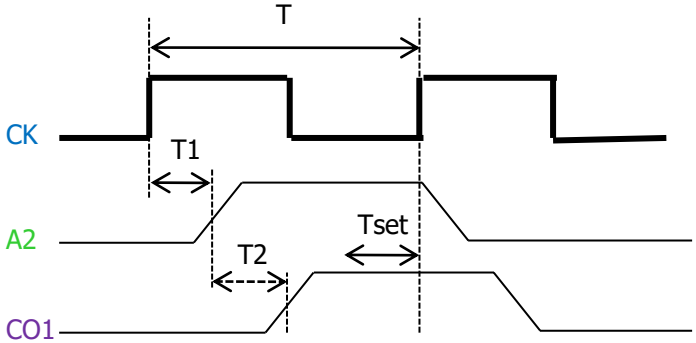
内　容

- セミナーの目的・目標
- デジタル回路の基礎
- スイッチレベル　デジタルCMOS回路
- デジタルCMOS回路の性能（消費電力、スピード）
- **同期回路設計とカウンタ回路**
- 加算器、ビットシフト、乗算器
- 事例1：　SAR ADC+分散型積和演算回路
- 事例2：　自己校正機能をもったTDC回路
- 事例3：　冗長アルゴリズムを用いたSAR ADC
- 最後に

同期回路とタイミング設計

Synchronous Circuit Design



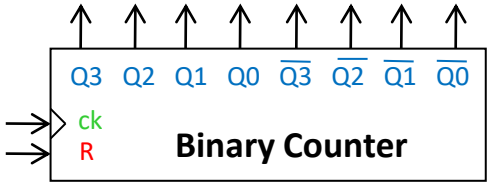


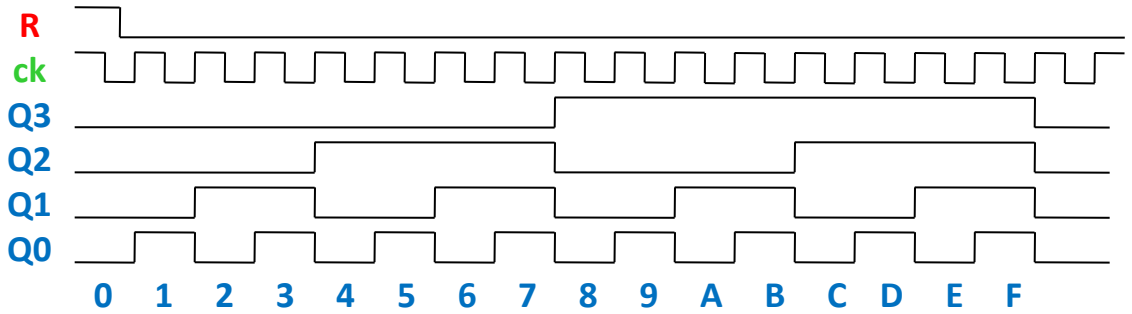
- Timing Requirement

T>T1+T2+Tset
T ： clock period
T1 ： Flip-Flop Delay
T2 ： Full Adder Delay
Tset ： Flip-Flop Setup Time

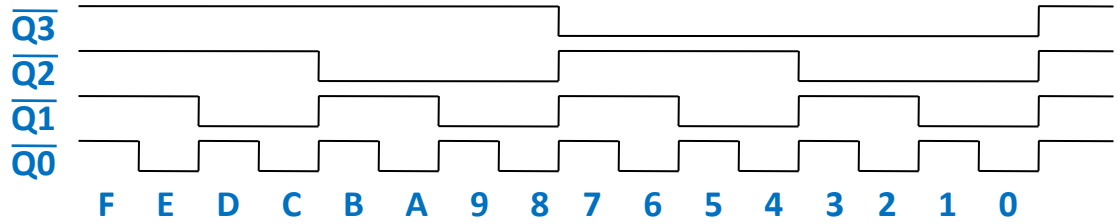
同期設計は
タイミング設計が
比較的容易

2進カウンタ (Binary Counter)





Up Counter

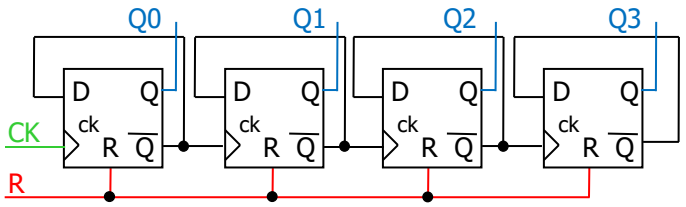


Down Counter

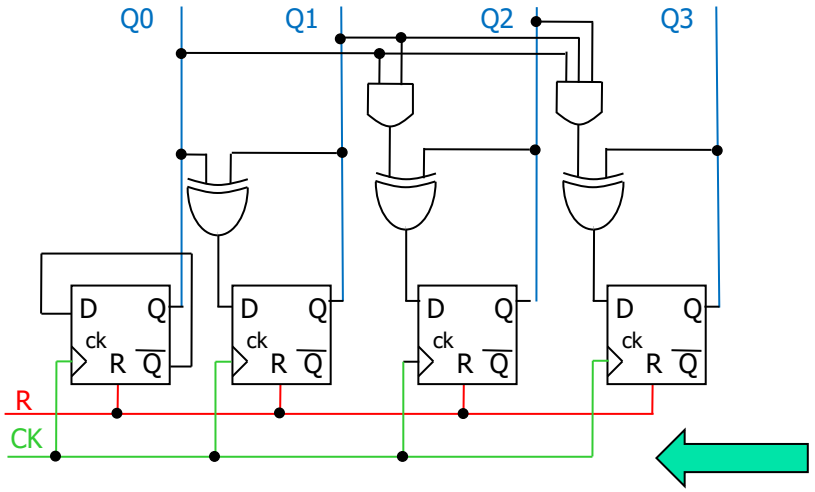
- クロック数を数える
- タイミング発生回路に使用される

非同期、同期カウンタ

非同期カウンタ



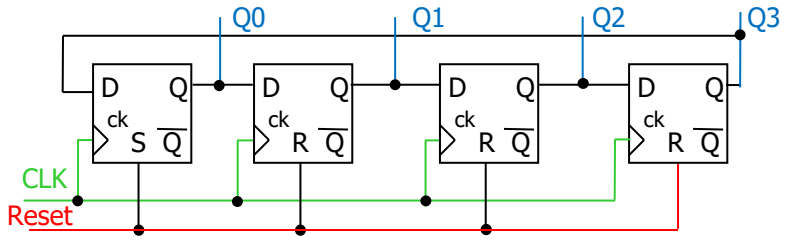
同期カウンタ

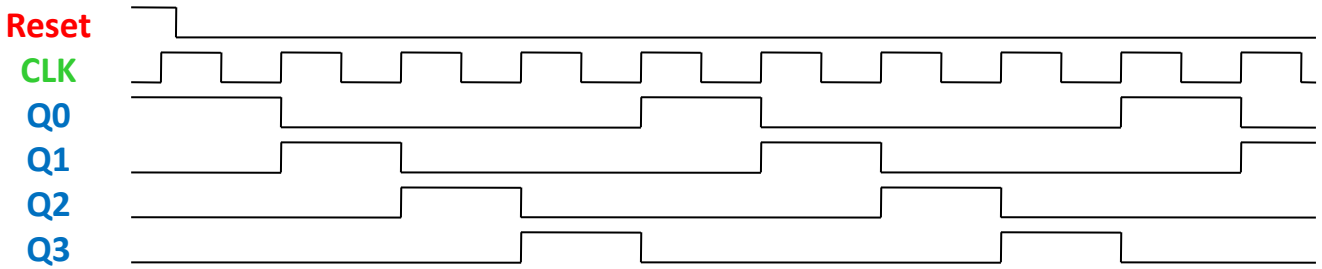


- 非同期回路は小規模になりえるが回路設計・変更・検証が難しい。
- デジタル回路は同期設計が基本

同期回路:
全てのフリップフロップのクロックが同一

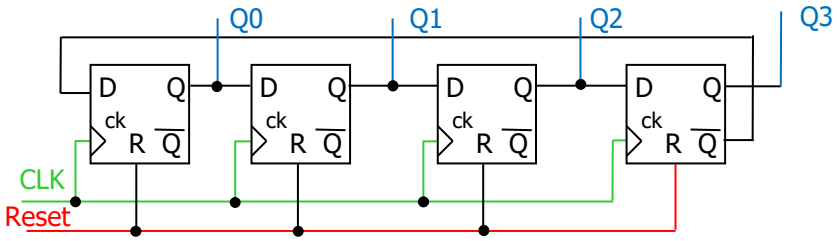
リング・カウンタ （Ring Counter)

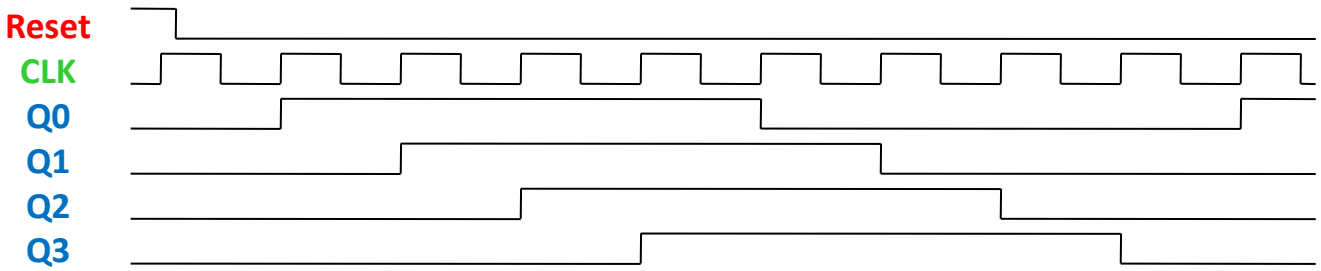




逐次比較近似ADCのタイミング発生回路等に使用される。

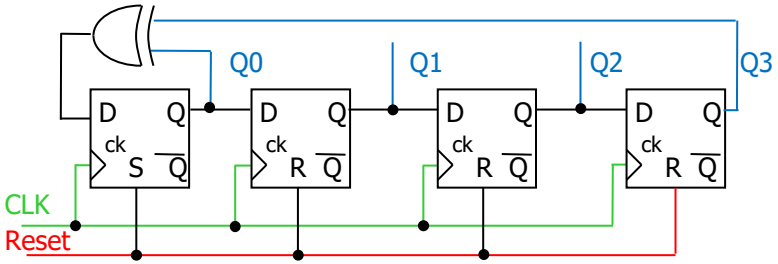
ジョンソン・カウンタ（Johnson Counter）

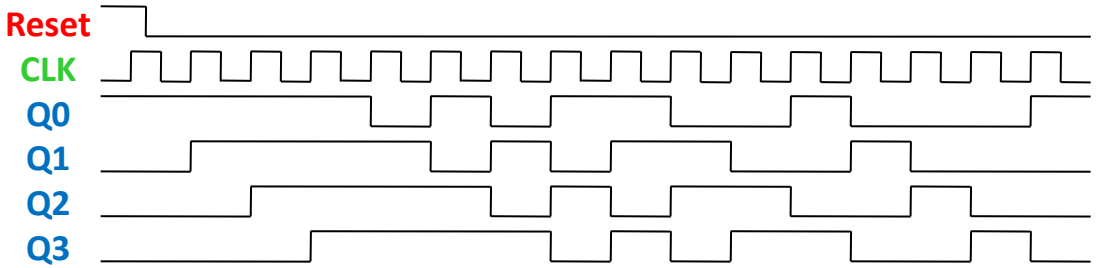




リングカウンタ回路と似ているが
出力信号は大きく異なる。
デジタル計算機のシーケンサ回路として使用された。

疑似ランダム信号発生回路 (Linear Feedback Shift Register)





- Q0=Q1=Q2=Q3=0 以外の15通りの信号を発生
- （再現性のある）疑似ランダム信号を発生 ⇨ M系列信号
- フィードバックの取り方には決まりあり

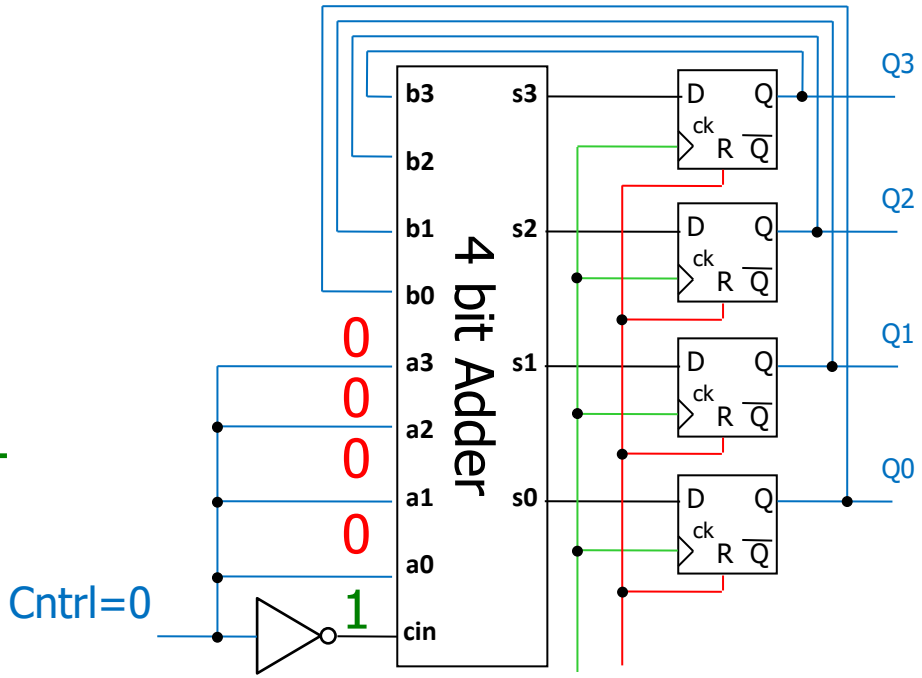
https://ja.wikipedia.org/wiki/線形帰還シフトレジスタ

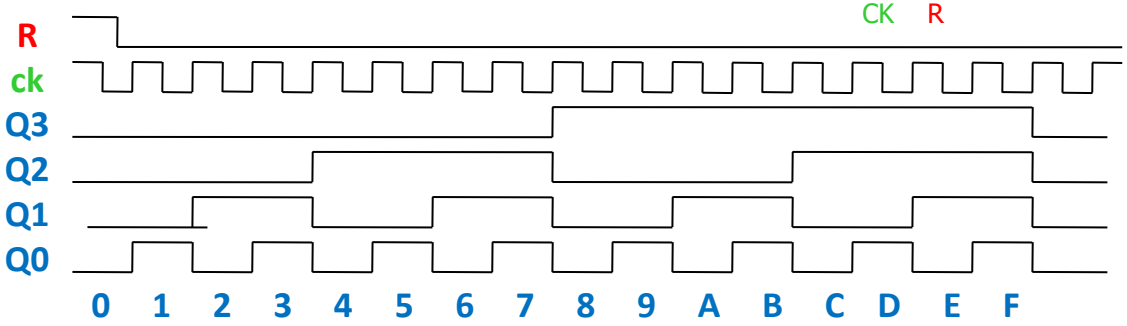
http://zakii.la.coocan.jp/signal/41_lfsr.htm

アップ・ダウン　2進カウンタ

制御信号
Cntrl=0 のとき
アップカウンタ

$$\overrightarrow{Q(n+1)} = \overrightarrow{Q(n)}+1$$

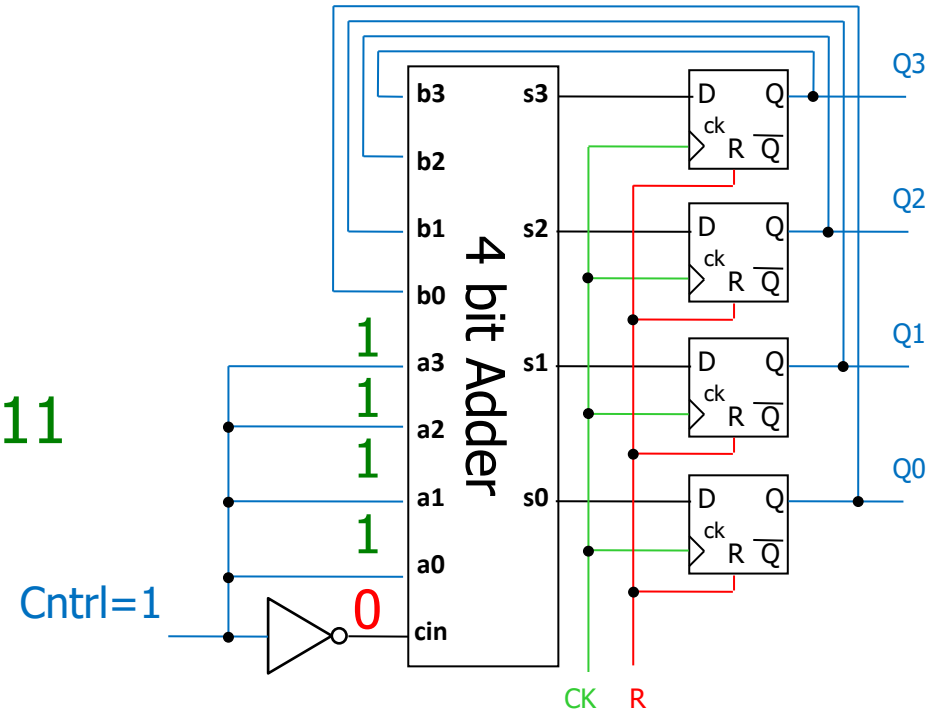


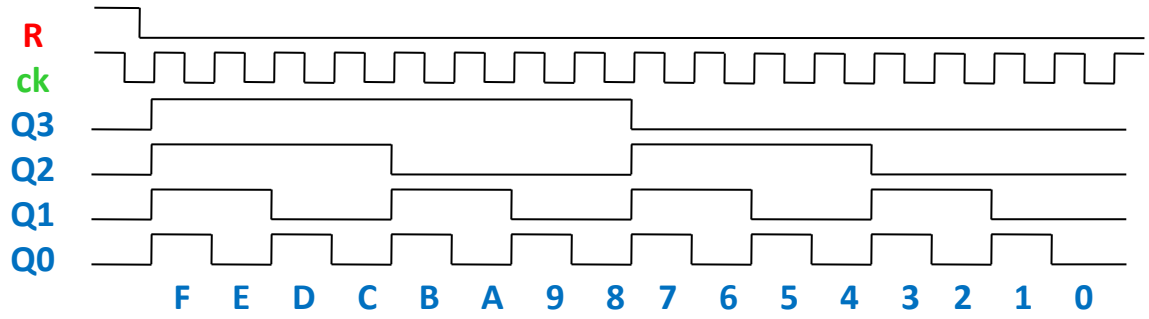


アップ・ダウン　2進カウンタ

制御信号
Cntrl=1 のとき
ダウンカウンタ

$$\overrightarrow{Q(n+1)} = \overrightarrow{Q(n)} + 1111$$
$$= \overrightarrow{Q(n)} - 1$$





内　容

- セミナーの目的・目標
- デジタル回路の基礎
- スイッチレベル　デジタルCMOS回路
- デジタルCMOS回路の性能（消費電力、スピード）
- 同期回路設計とカウンタ回路
- **加算器、ビットシフト、乗算器**
- 事例1：　SAR ADC+分散型積和演算回路
- 事例2：　自己校正機能をもったTDC回路
- 事例3：　冗長アルゴリズムを用いたSAR ADC
- 最後に

デジタル加算

2進数の加算

```
  0011   (3)
+) 1011  (11)
------------
  1110   (14)
```

10進数の加算

```
  437
+)258
------
  695
```

2入力2進加算

0+0=00

0+1=01

1+0=01

1+1=10

3入力2進加算

0+0+0=00

0+0+1=01

0+1+1=10

1+1+1=11

デジタル加算器の実現(2)
（全加算器； Full Adder）

3入力2進加算

A　入力1

B　入力2

+) Cin　下からの繰り上がり

Co S

$S = A \oplus B \oplus Cin$

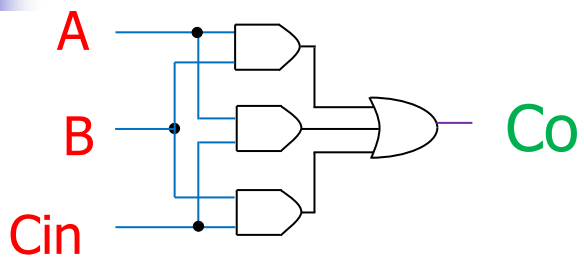
$Co = B \cdot Cin + A \cdot Cin + A \cdot B$

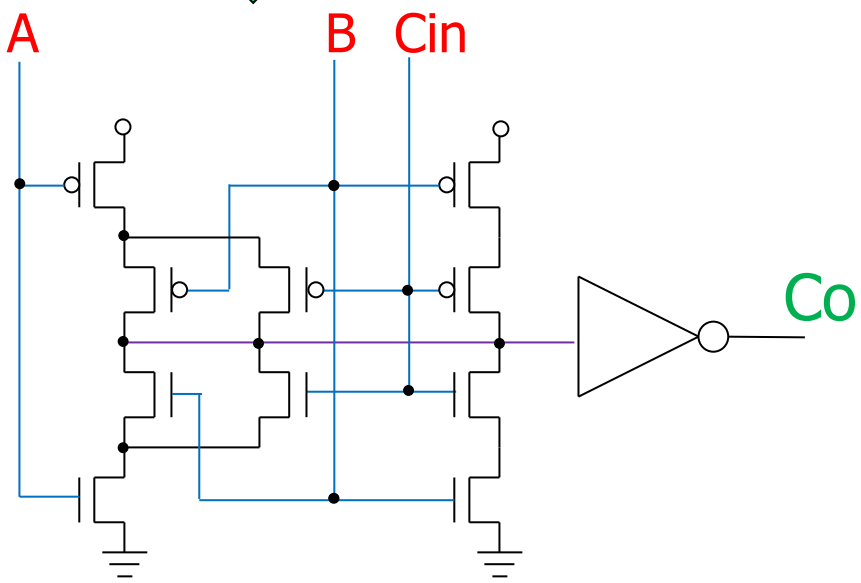
(Co は A, B, Cin の多数決）

真理値表

A	B	Cin	Co	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
1	0	0	0	1
0	1	1	1	0
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

多数決回路 (Majority Circuit)





真理値表

A	B	Cin	Co	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
1	0	0	0	1
0	1	1	1	0
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

全加算器 (Full Adder)の補足説明

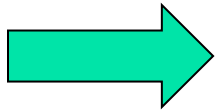
Cin:　Carry in (下位の桁からの繰り上げ)

S:　　Sum（加算結果）

Cout:　Carry out （上位の桁への繰り上げ）

```
    0   1
+)  1   1
-------
  1 0   0
```

を考える。



全加算器の演算

```
  1
  0   1
+)1   1
-------
1 0   0
```

全加算器の補足説明(続き)

```
    0  1  0  1  (5)
+)  0  1  1  1  (7)
-------------------
 0  1  1  0  0  (12)
```

を考える。

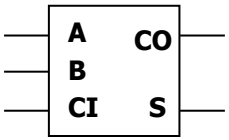
```
    1  1  1
    0  1  0  1
+)  0  1  1  1
-------------------
 0  1  1  0  0
```

Carry（桁上げ）

Sum（加算結果）

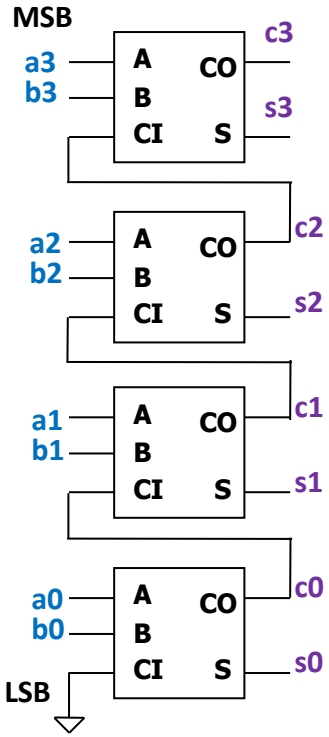
全加算器と桁上げ伝播

Full Adder



A	B	CI	S	CO
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
1	0	0	1	0
0	1	1	0	1
1	1	0	0	1
1	0	1	0	1
1	1	1	1	1

Adder & Carry Propagation



Example :

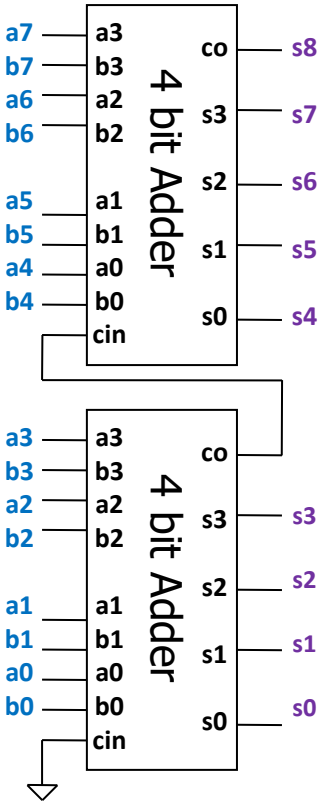
$$\begin{array}{r} 0\,1\,1\,1 \\ +\ 0\,1\,0\,1 \\ \hline 1\,1\,0\,0 \end{array}$$

(a3 a2 a1 a0) =(0 1 1 1)
(b3 b2 b1 b0)=(0 1 0 1)

(s3 s2 s1 s0)=(1 1 0 0)
(c3 c2 c1 c0)=(0 1 1 1)

桁上げ選択加算器 (Carry Select Adder) による高速化

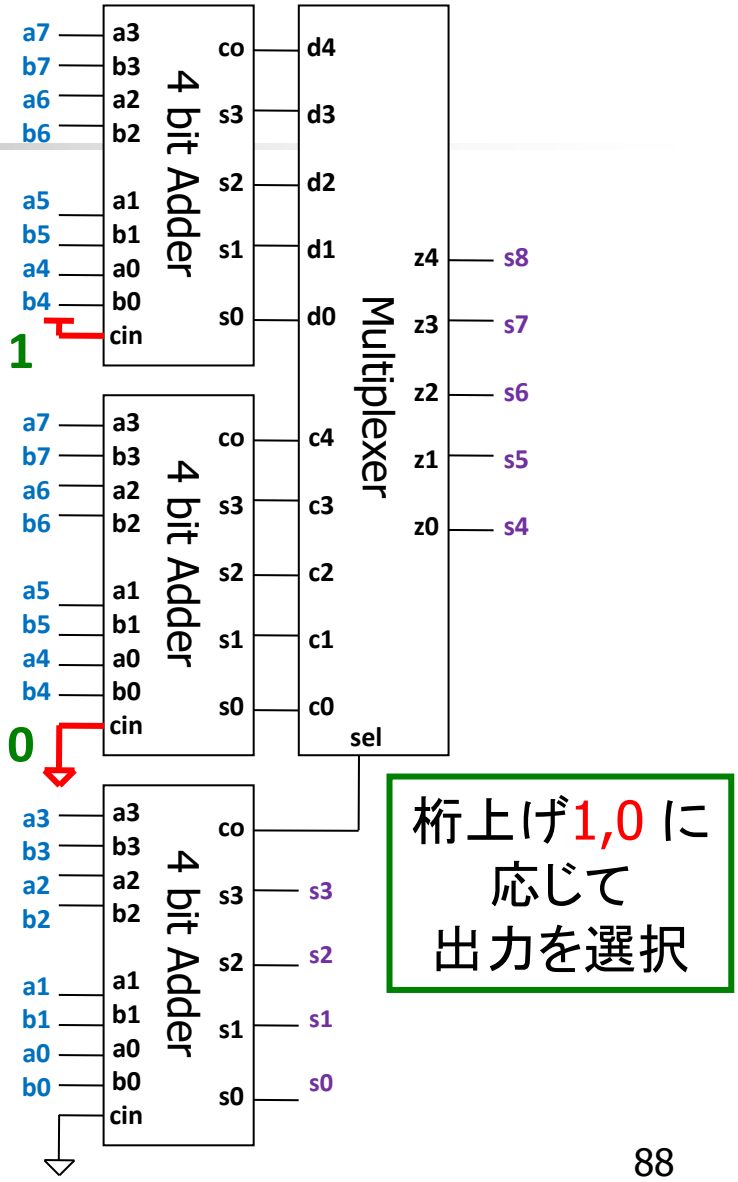
通常の
8bit 加算器



桁上げ1の場合を計算

桁上げ0の場合を計算

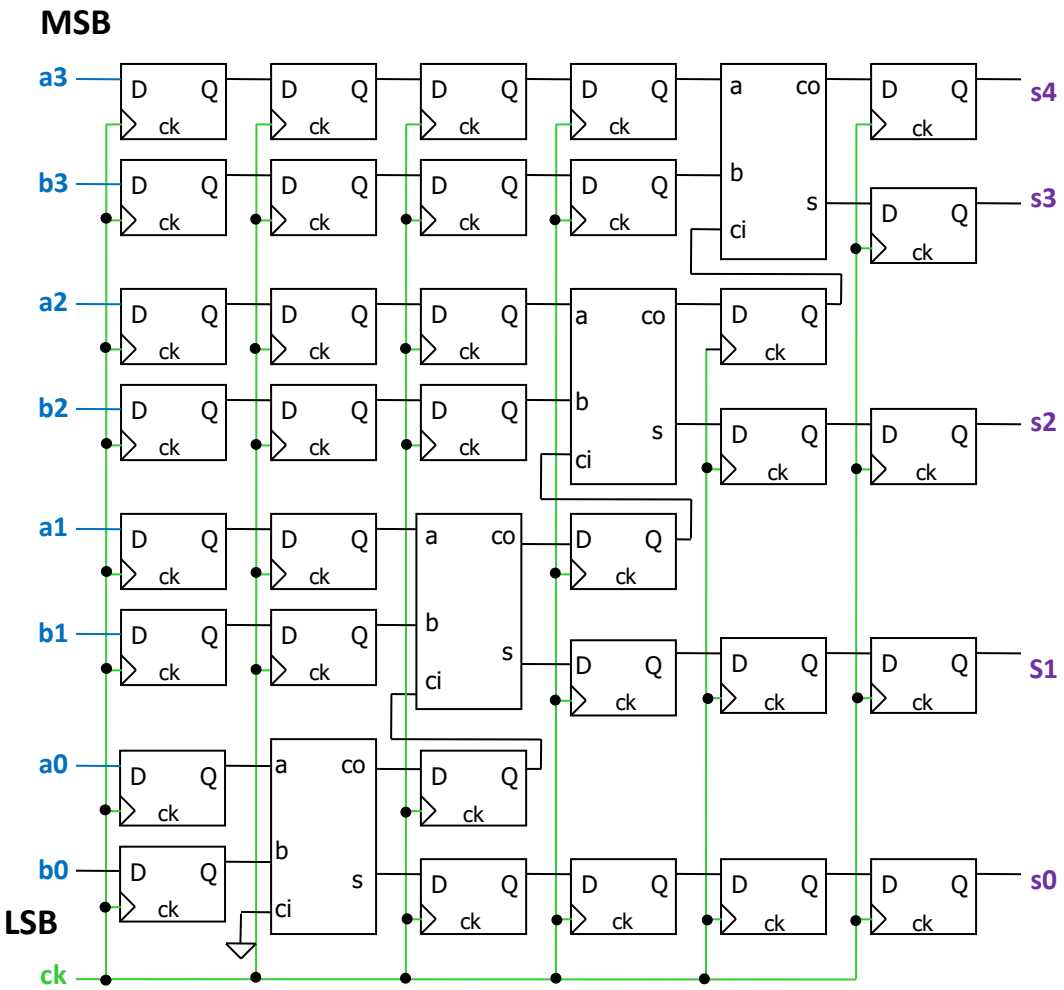
約1．5倍の回路規模で約2倍のスピード



桁上げ1,0 に応じて出力を選択

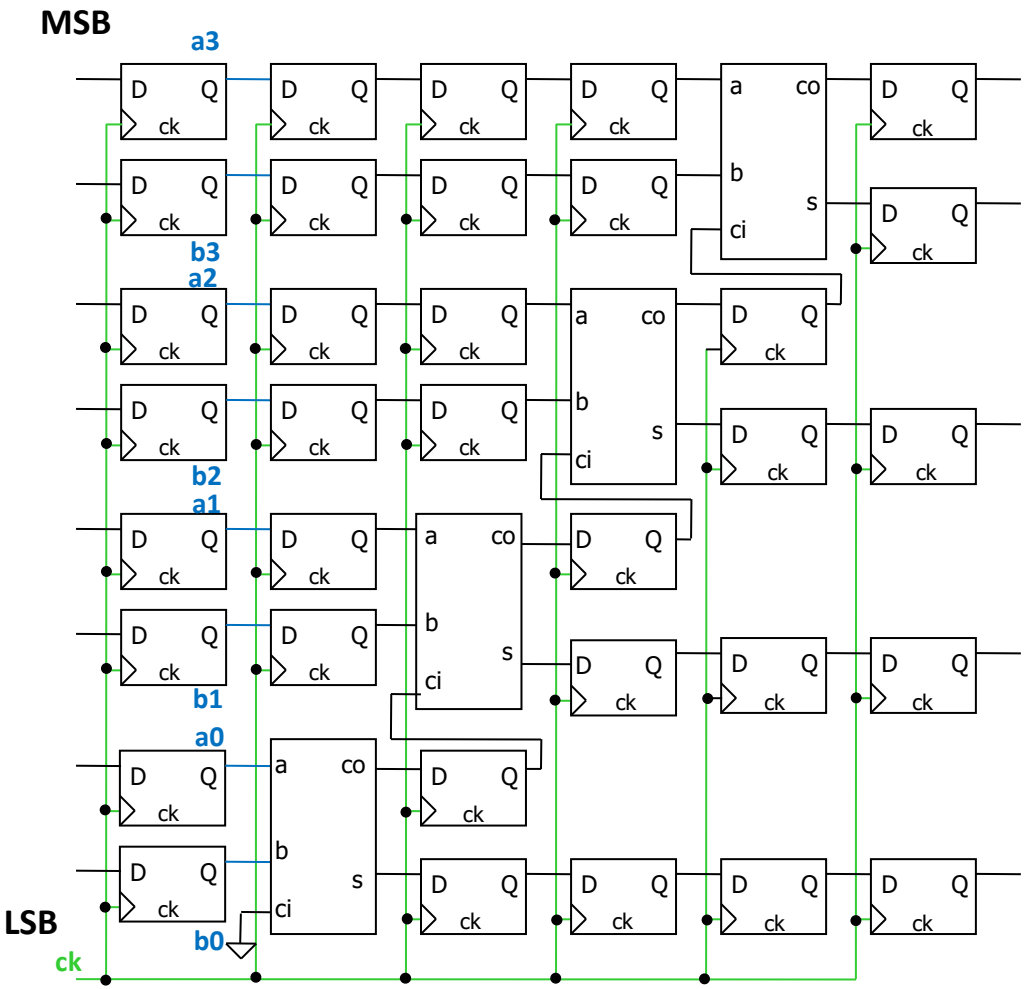
パイプライン加算器の構成

初期状態



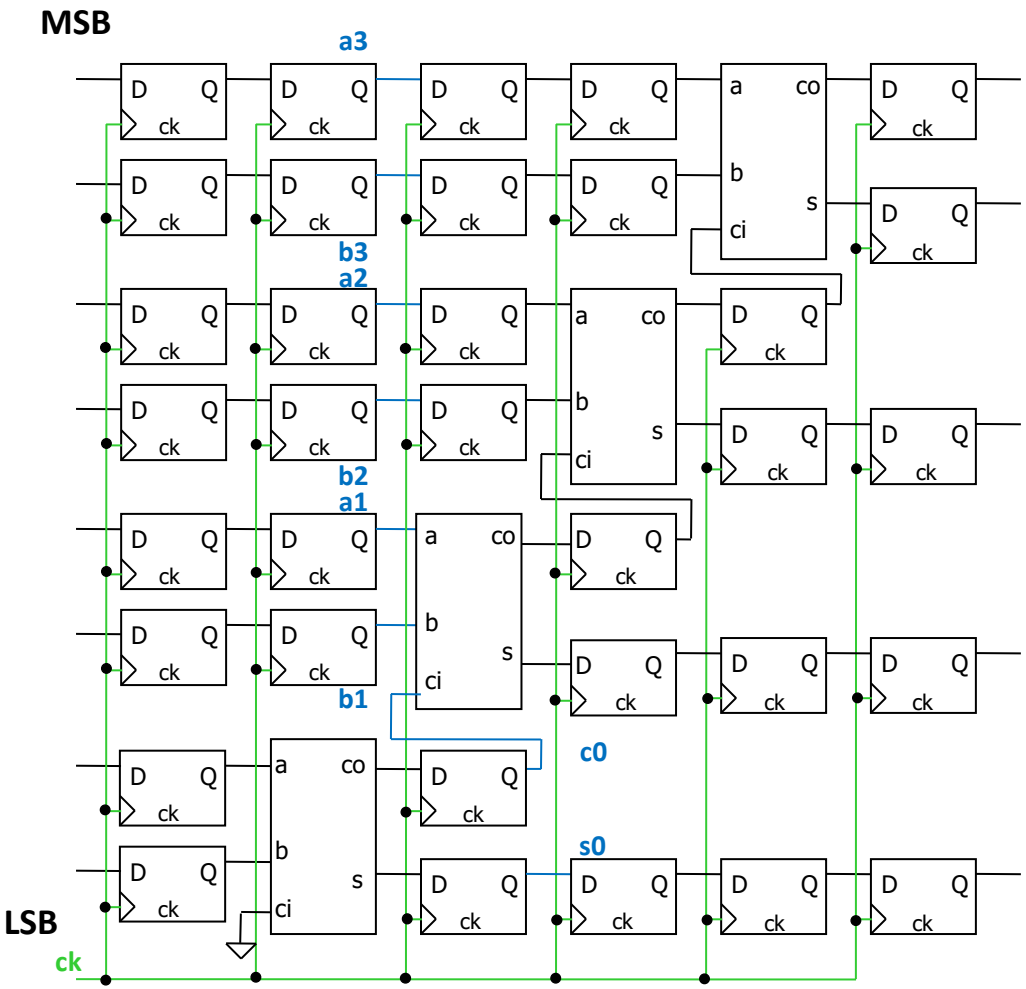
パイプライン加算器の動作

1クロック後



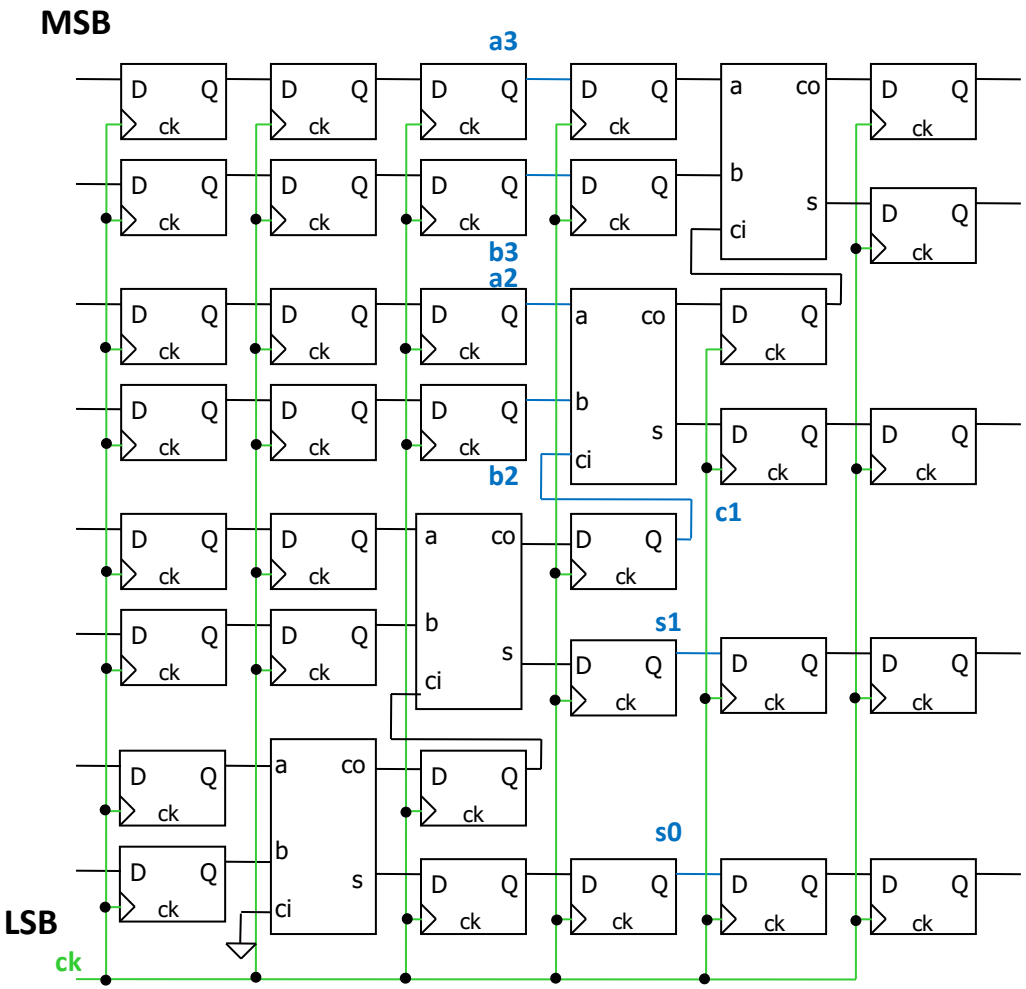
パイプライン加算器の動作

2クロック後



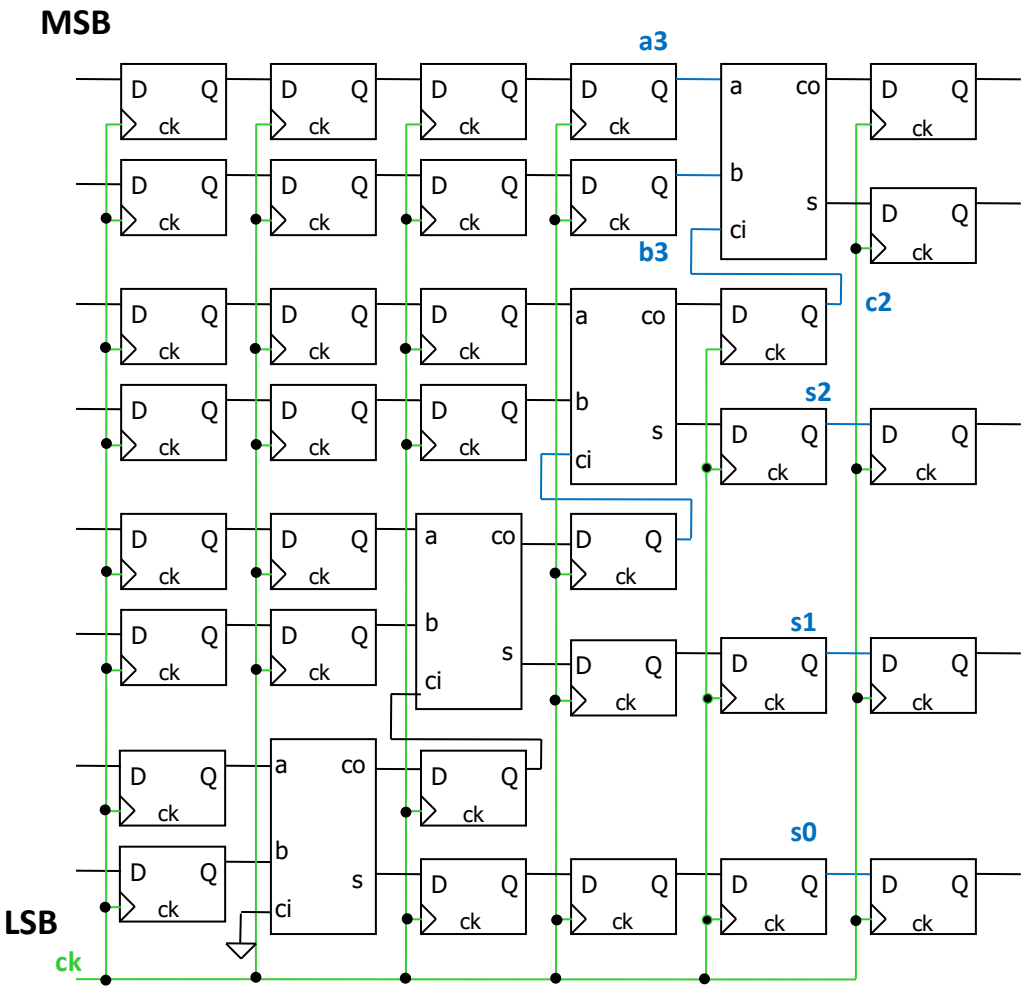
パイプライン加算器の動作

3クロック後



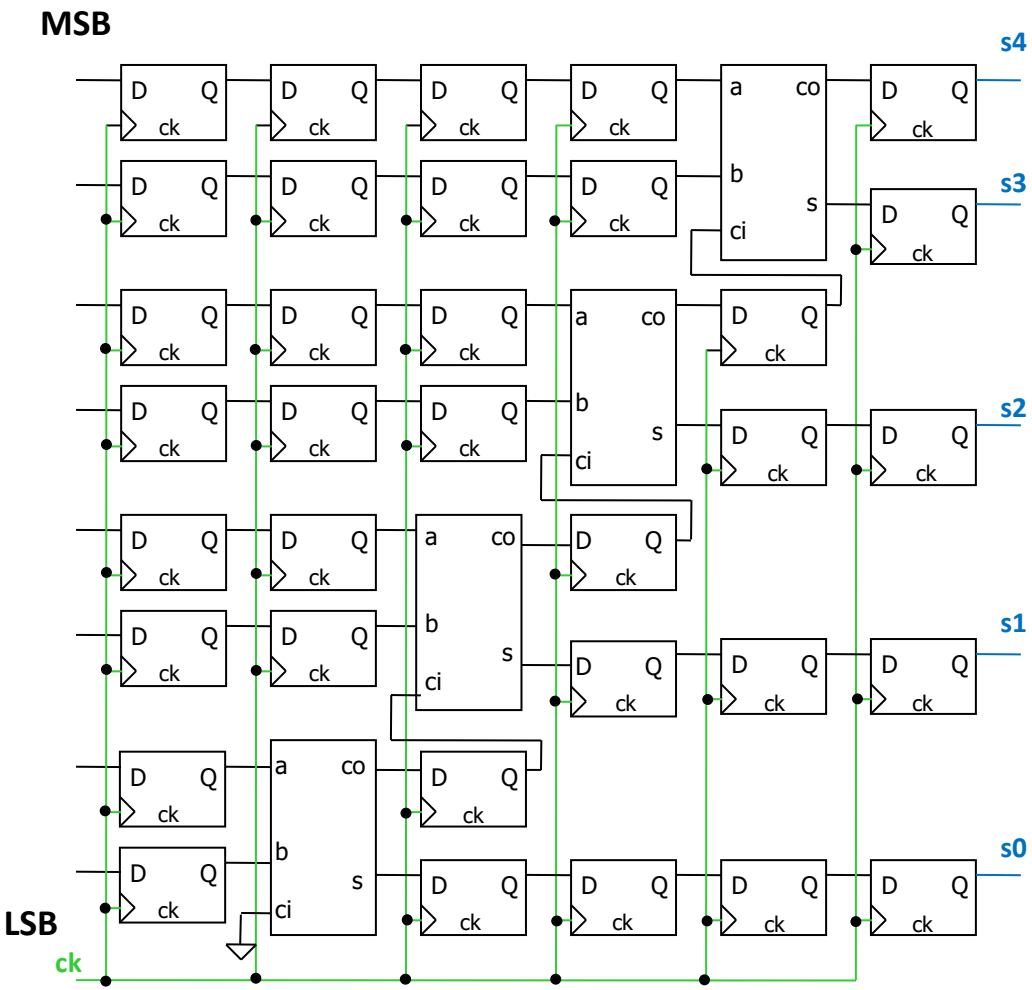
パイプライン加算器の動作

4クロック後



パイプライン加算器の動作

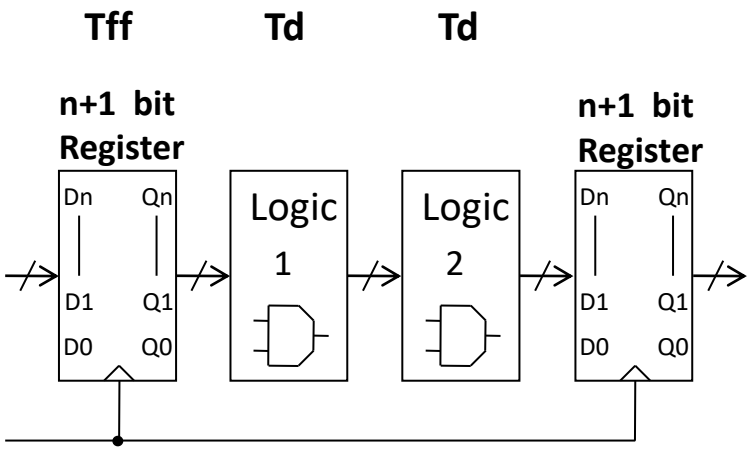
5クロック後



パイプライン構成による高速化

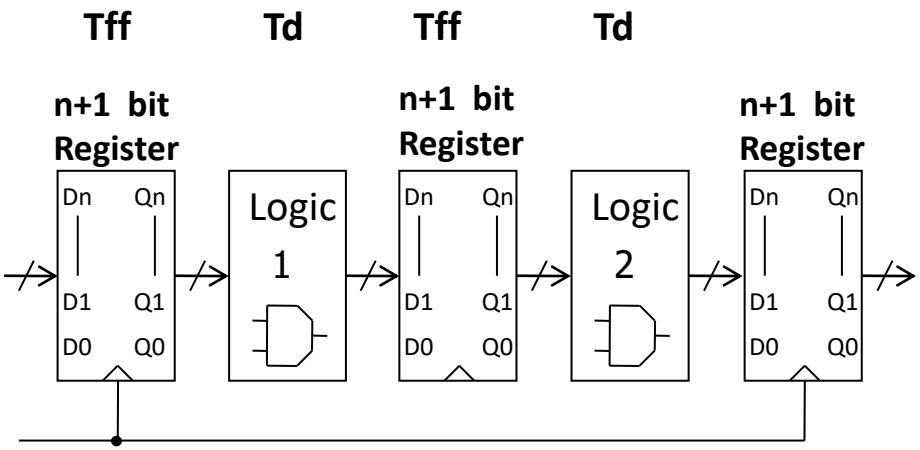
Pipeline Circuit

● **Circuit 1**



clock period T > Tff + 2Td + Tset

● **Circuit 2**



clock period T > Tff + Td + Tset

Td >> Tff + Tset ⇒ circuit 2 は circuit 1 の2倍高速

パイプライン構成の特徴

- バケツリレー　の原理
- レジスタを組み合せ回路の中間にいれればよい
- ２つの性能指標
  Throughput：出力が出てくるレート（１クロック）
  Latency：　ある入力の結果が出力されるまでの遅延　（５クロック）
- パイプライン構成では
  Throughput は良くなる
  Latency は数クロックかかる（良くない）
- フィードバック構成の使用は注意必要

桁上げ計算節約加算器
Carry Save Adder

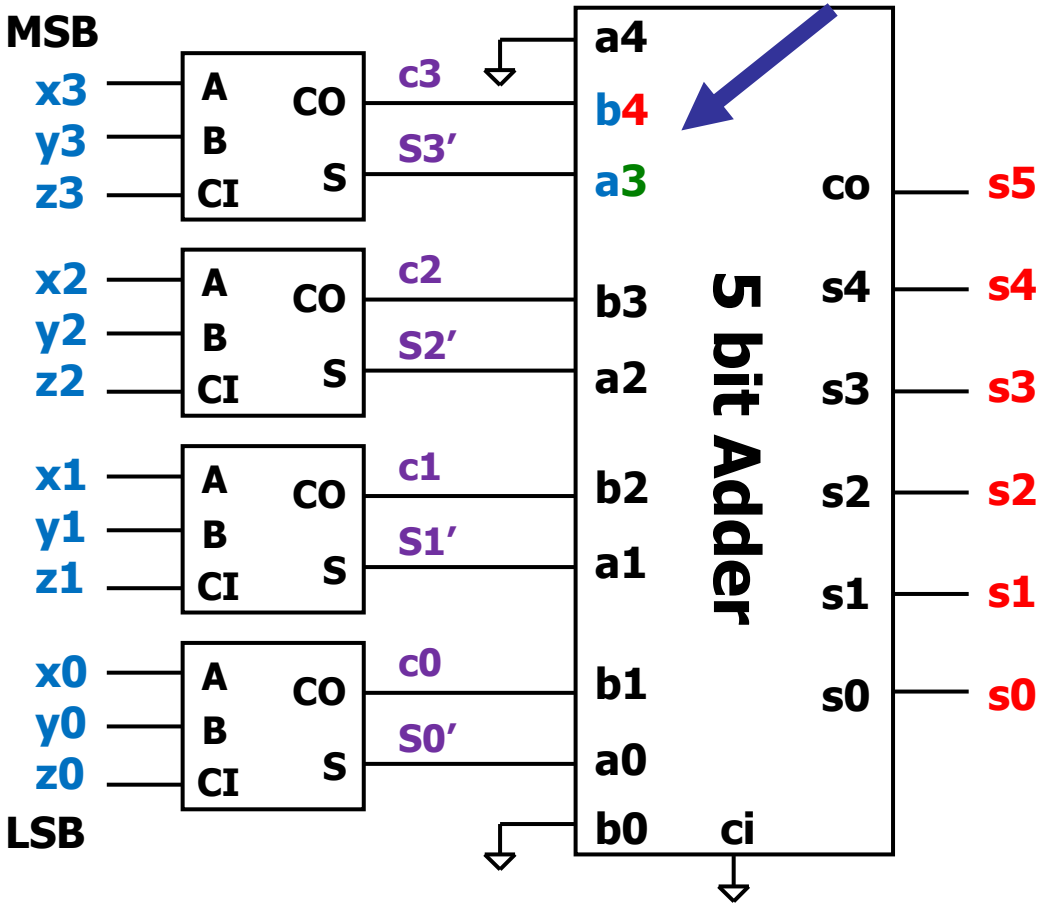
co は bn+1
s は an
に入力

3入力加算器　　S = X + Y + Z

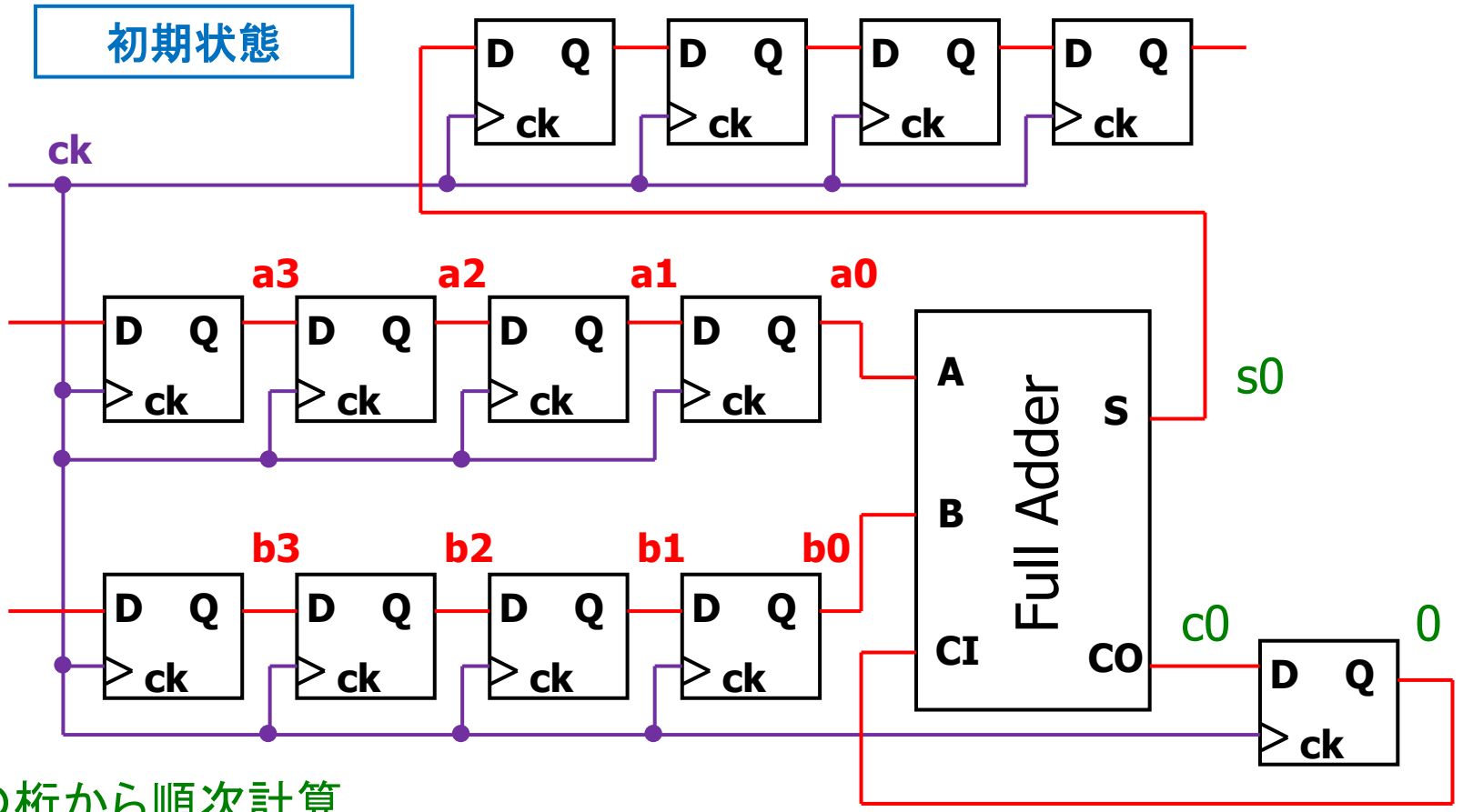
- 多入力加算
- 最後に1回だけ桁上げ計算

↓

- 高速
- 回路規模　小

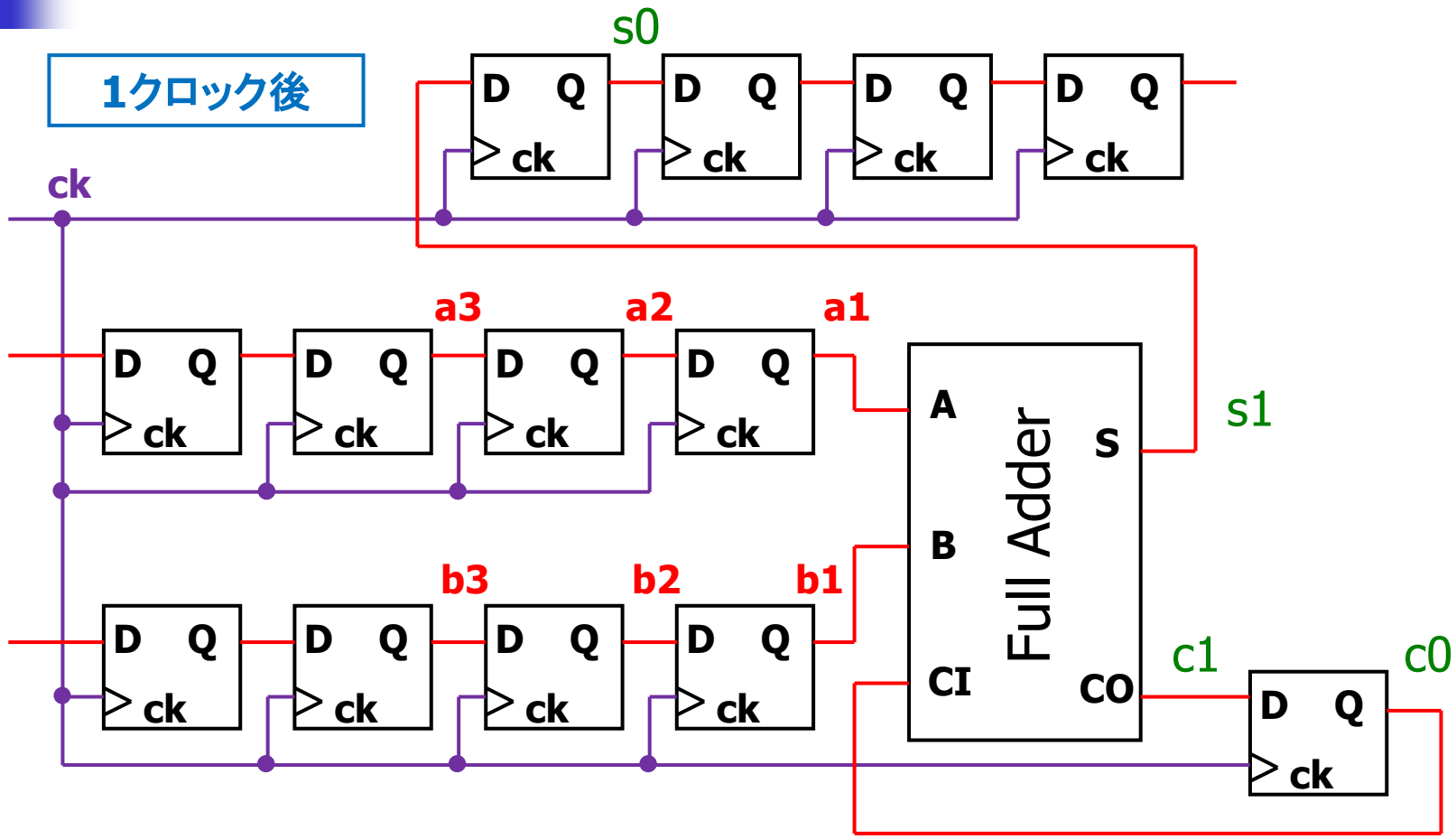


ビットシリアル加算器の構成
全加算器は1個でよい

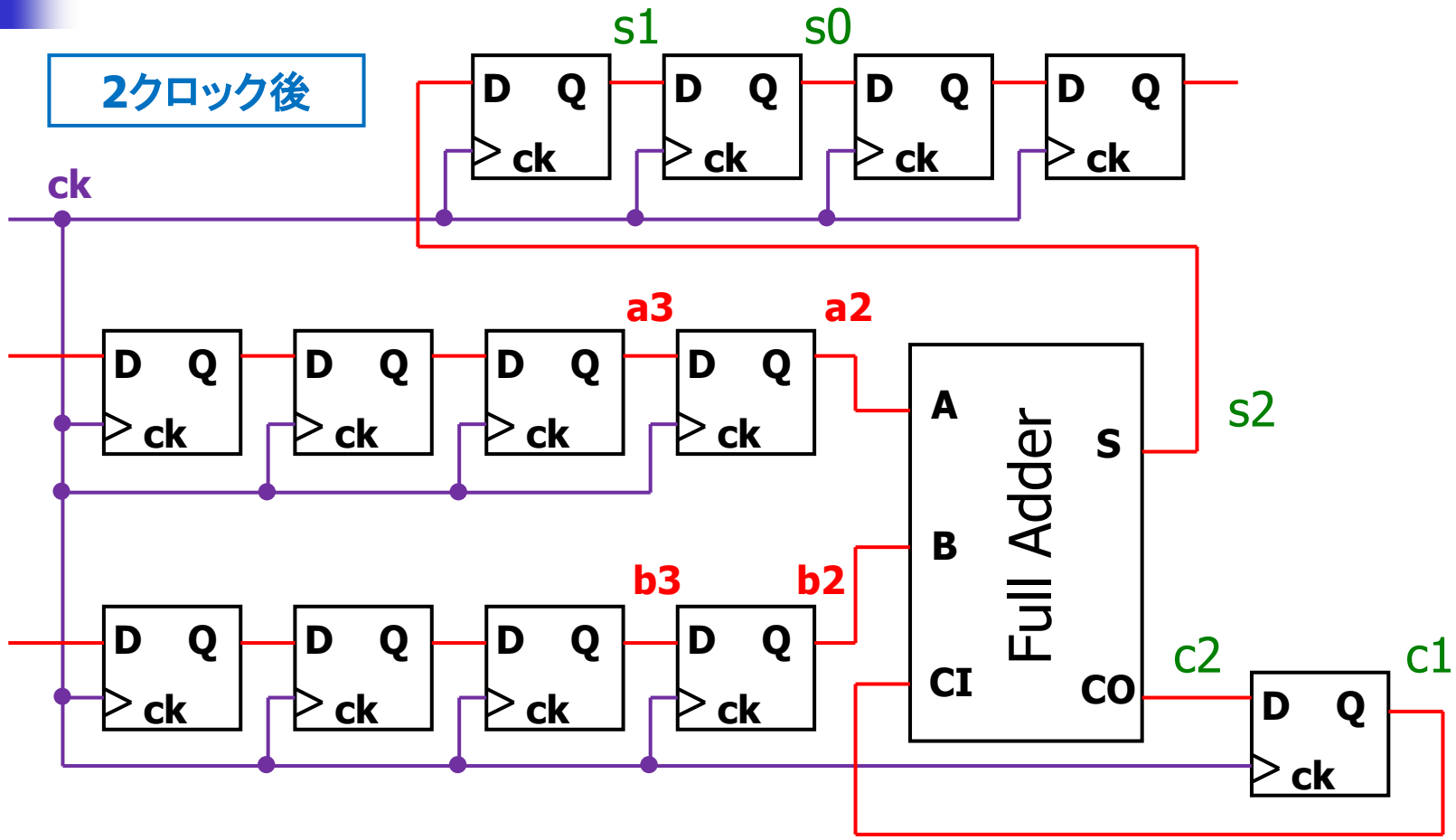


下の桁から順次計算
加算演算に桁数だけクロック数が必要

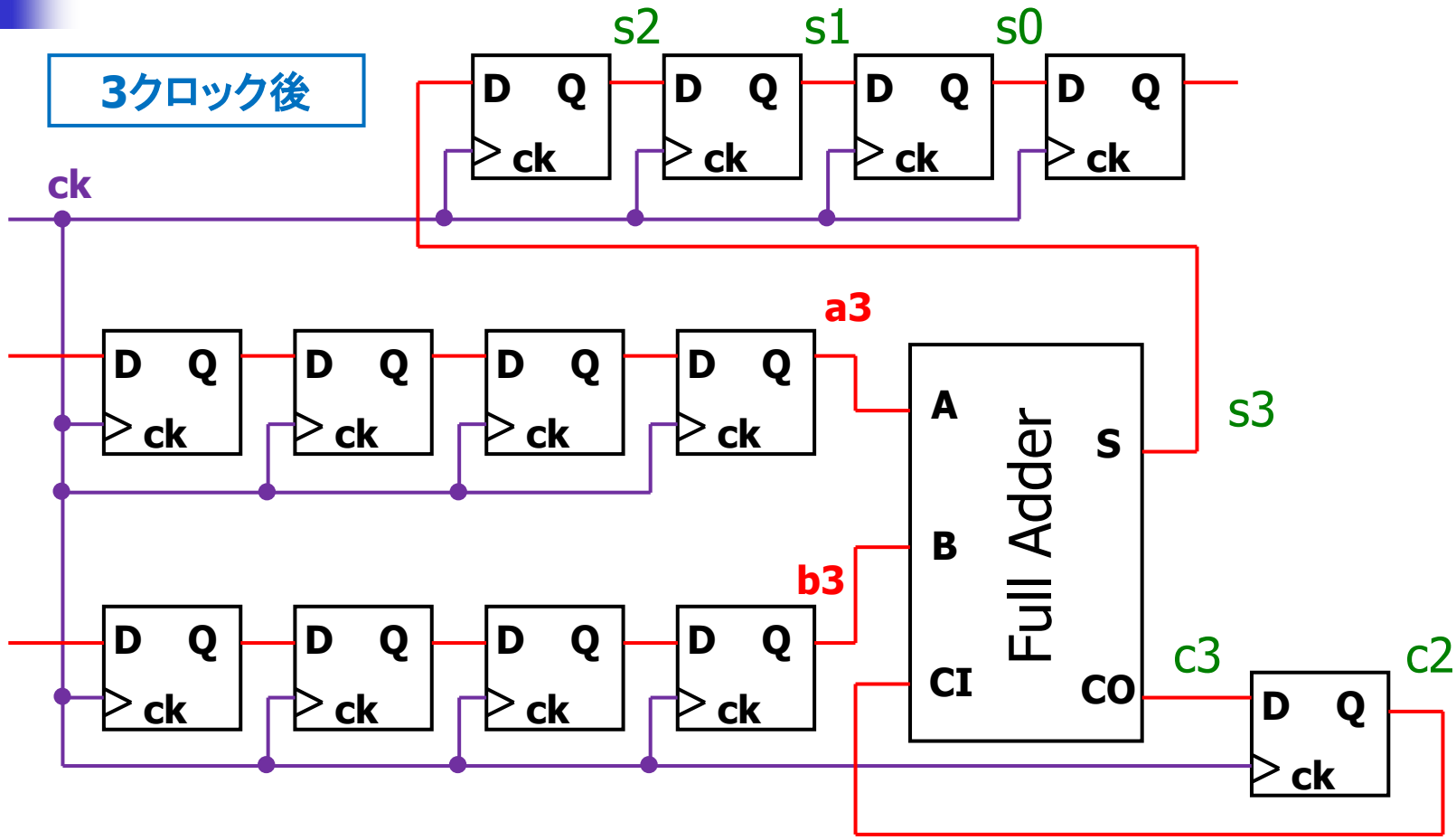
ビットシリアル加算器の動作



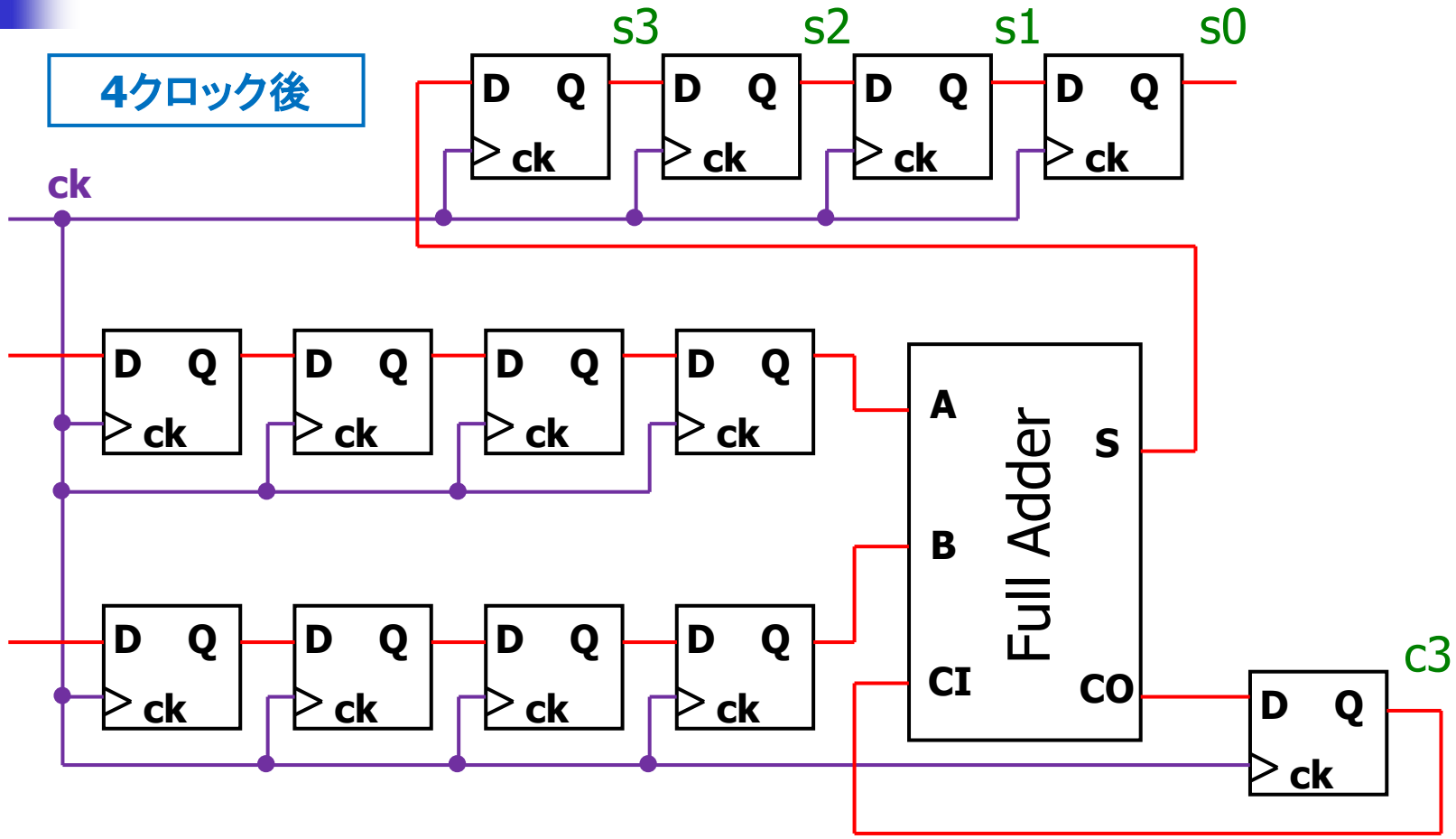
ビットシリアル加算器の動作



ビットシリアル加算器の動作



ビットシリアル加算器の動作



ビットシフトと乗算

10進数で　x10, x100, ÷1000 等は簡単
同様に2進数で x2, x4、÷8 等は簡単

● 1 bit left shift ⟷ ×2

b4 b3 b2 b1 b0

0 0 0 1 1 → 3

0 0 1 1 0 → 3 ×2=6

0 1 1 0 0 → 6 ×2=12

1 1 1 0 1 → -3

1 1 0 1 0 → -3 ×2=-6

1 0 1 0 0 → -6 ×2=-12

●1 bit right shift ⟷ ÷ 2

b4 b3 b2 b1 b0

0 1 0 0 0 → 8

0 0 1 0 0 → 8 / 2=4

0 0 0 1 0 → 4 / 2=2

1 1 0 0 0 → -8

1 1 1 0 0 → -8 / 2=-4

1 1 1 1 0 → -4 / 2=-2

b4	b3	b2	b1	b0	十進数
0	0	0	0	0	0
0	0	0	0	1	1
0	0	0	1	0	2
0	0	0	1	1	3
0	0	1	0	0	4
0	0	1	0	1	5
0	0	1	1	0	6
0	0	1	1	1	7
0	1	0	0	0	8
0	1	0	0	1	9
0	1	0	1	0	10
0	1	0	1	1	11
0	1	1	0	0	12
0	1	1	0	1	13
0	1	1	1	0	14
0	1	1	1	1	15
1	0	0	0	0	-16
1	0	0	0	1	-15
1	0	0	1	0	-14
1	0	0	1	1	-13
1	0	1	0	0	-12
1	0	1	0	1	-11
1	0	1	1	0	-10
1	0	1	1	1	-9
1	1	0	0	0	-8
1	1	0	0	1	-7
1	1	0	1	0	-6
1	1	0	1	1	-5
1	1	1	0	0	-4
1	1	1	0	1	-3
1	1	1	1	0	-2
1	1	1	1	1	-1

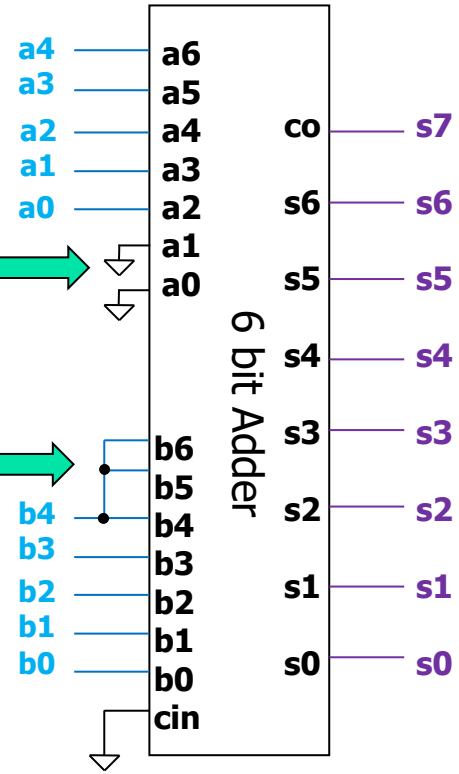
ビットシフトで2^n 倍乗算を実現

S= 4×A+B

乗算器を
使用せずに実現

2ビット
左シフトで
X4 を実現

2の補数表現で
符号ビットの拡張



デジタル乗算

2進数の乗算

```
   0101   (5)
X) 1011   (11)
-------------
   0101
  0101
 0000
0101
-------------
0110111   (55)
```

10進数の乗算

```
    437
X)  258
-------
   3496
  2185
   874
-------
112746
```

2進デジタル乗算器

$$X = \sum_{i=0}^{m-1} X_i 2^i$$

$$Y = \sum_{j=0}^{n-1} Y_j 2^j$$

$$P = X \times Y = \sum_{i=0}^{m-1} X_i 2^i \bullet \sum_{j=0}^{n-1} Y_j 2^j$$

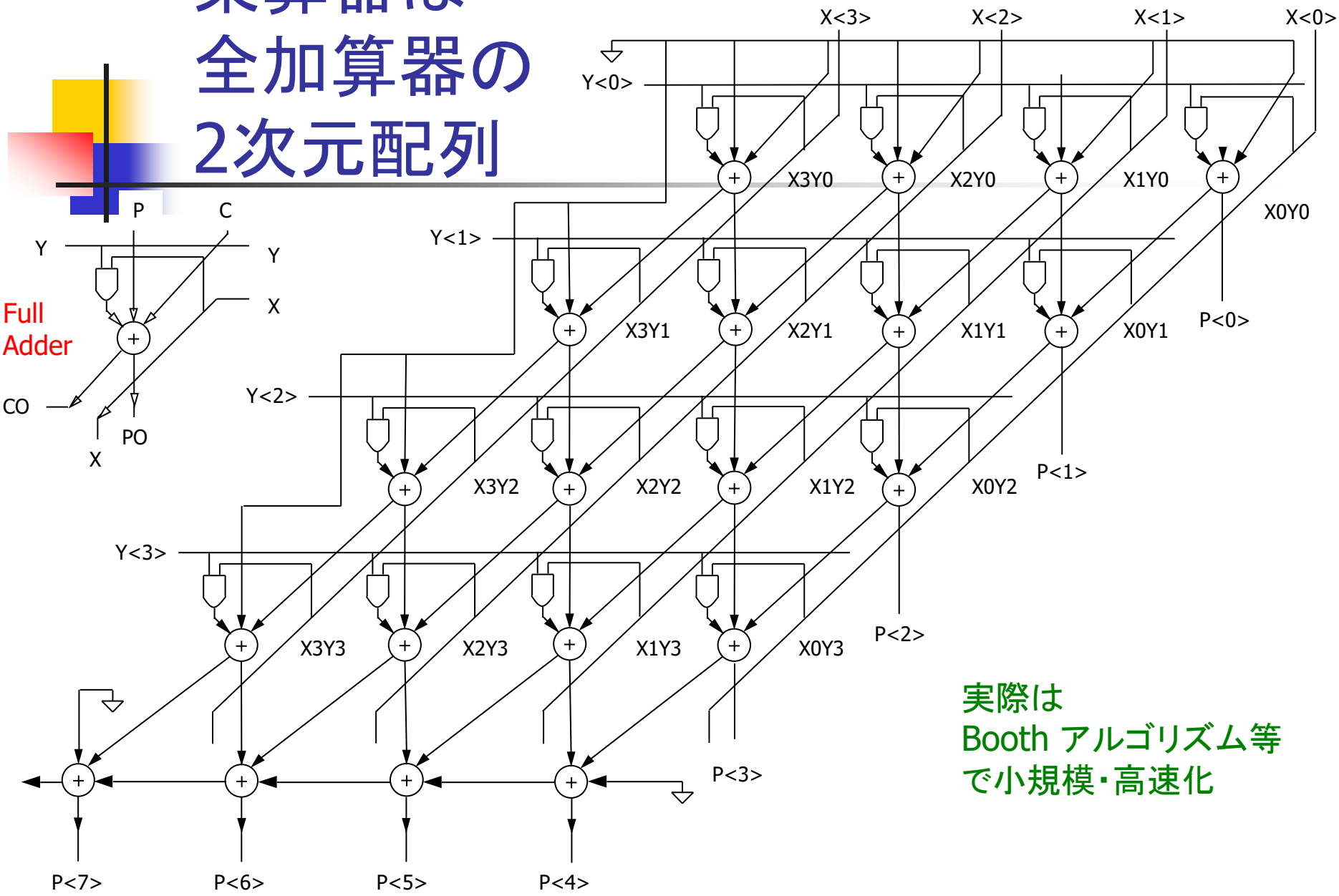
$$= \sum_{i=0}^{m-1} \bullet \sum_{j=0}^{n-1} Y_j 2^j \left(X_i Y_j\right) 2^{i+j}$$

$$= \sum_{k=0}^{m+n-1} P_k 2^k$$

4-bit Multiplier Partial Products

				X3	X2	X1	X0	Multiplicand
				Y3	Y2	Y1	Y0	Multiplier
				X3Y0	X2Y0	X1Y0	X0Y0	
			X3Y1	X2Y1	X1Y1	X0Y1		
		X3Y2	X2Y2	X1Y2	X0Y2			
	X3Y3	X2Y3	X1Y3	X0Y3				
P7	P6	P5	P4	P3	P2	P1	P0	Product

乗算器は全加算器の2次元配列



実際は
Booth アルゴリズム等
で小規模・高速化

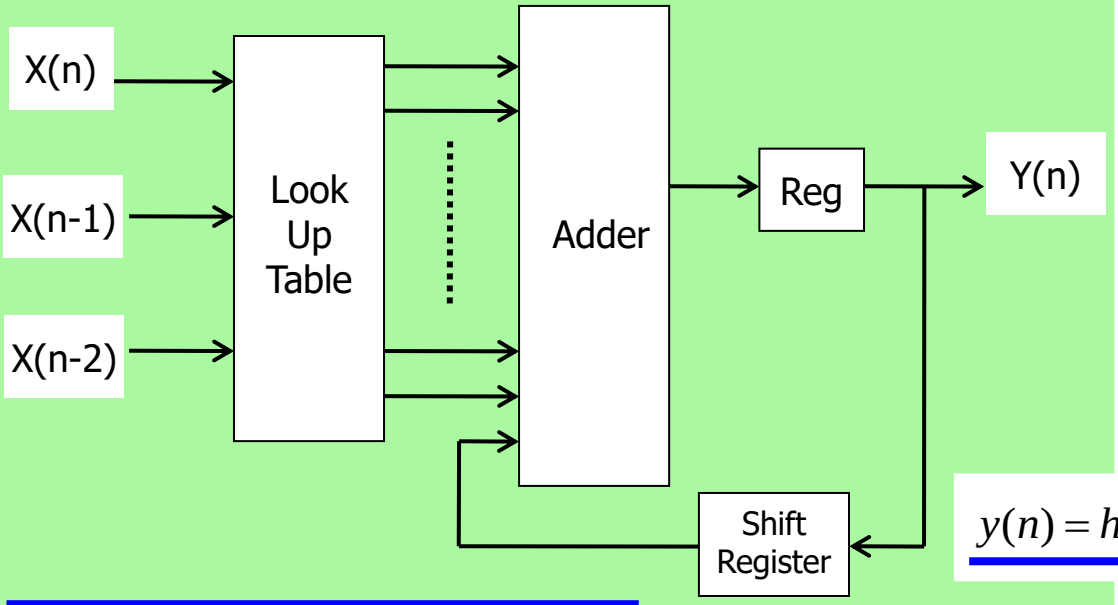
内　容

- セミナーの目的・目標
- デジタル回路の基礎
- スイッチレベル　デジタルCMOS回路
- デジタルCMOS回路の性能（消費電力、スピード）
- 同期回路設計とカウンタ回路
- 加算器、ビットシフト、乗算器
- **事例1：　SAR ADC+分散型積和演算回路**
- 事例2：　自己校正機能をもったTDC回路
- 事例3：　冗長アルゴリズムを用いたSAR ADC
- 最後に

乗算器を使用しない
定係数積和演算回路

定係数の内積演算をルックアップ・テーブル（ROM等）により実現する計算手法

分散型積和演算



分散型積和演算回路構成

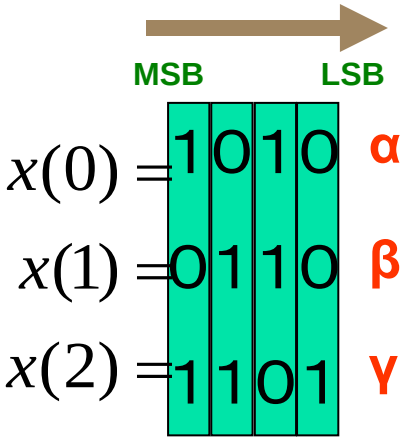
上位ビットからのシリアル演算

$$y(n) = h_0 x(n) + h_1 x(n-1) + h_0 x(n-2)$$

デジタル入力

分散型積和演算　動作原理

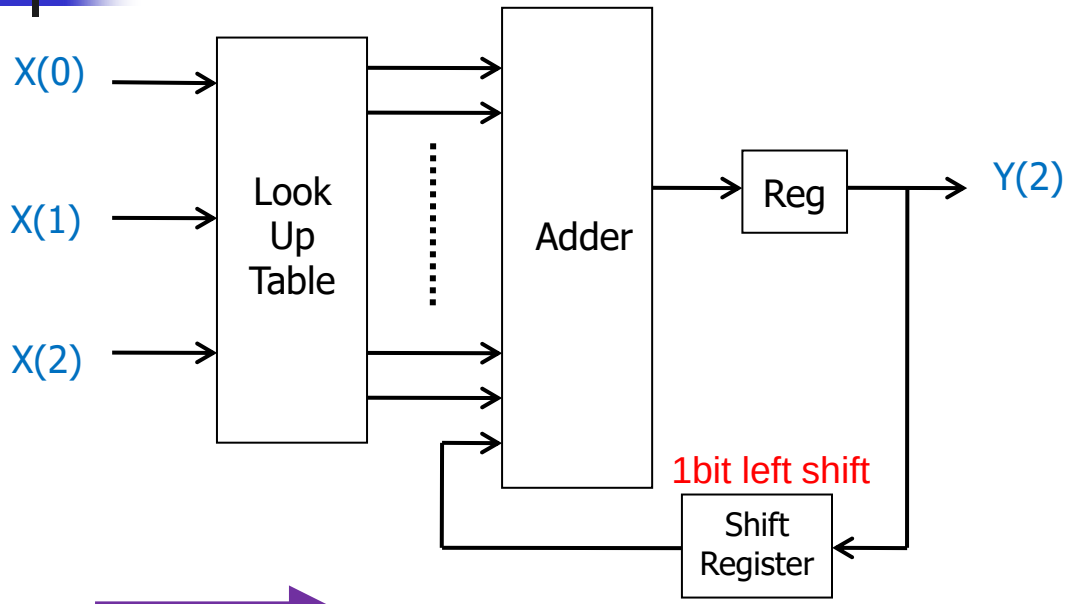
$$y(n) = h_0 x(n) + h_1 x(n-1) + h_0 x(n-2)$$



Look Up Table

γ	β	α	Y(n)
0	0	0	0
0	0	1	h0
0	1	0	h1
0	1	1	h0+h1
1	0	0	h0
1	0	1	2*h0
1	1	0	h1+h0
1	1	1	2*h0+h1

分散型積和演算　動作原理



MSB　LSB

x(0)=1 0 1 0

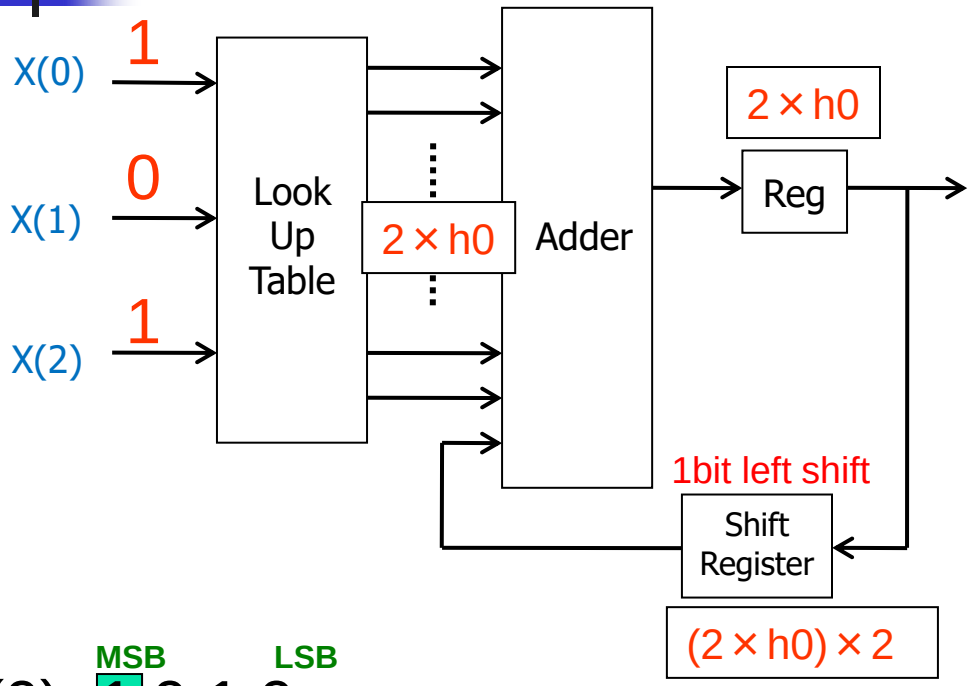
x(1)=0 1 1 0

x(2)=1 1 0 1

右シフトに処理

γ	β	α	Y(n)
0	0	0	0
0	0	1	h0
0	1	0	h1
0	1	1	h0+h1
1	0	0	h0
1	0	1	2*h0
1	1	0	h1+h0
1	1	1	2*h0+h1

分散型積和演算　動作原理



MSB　LSB

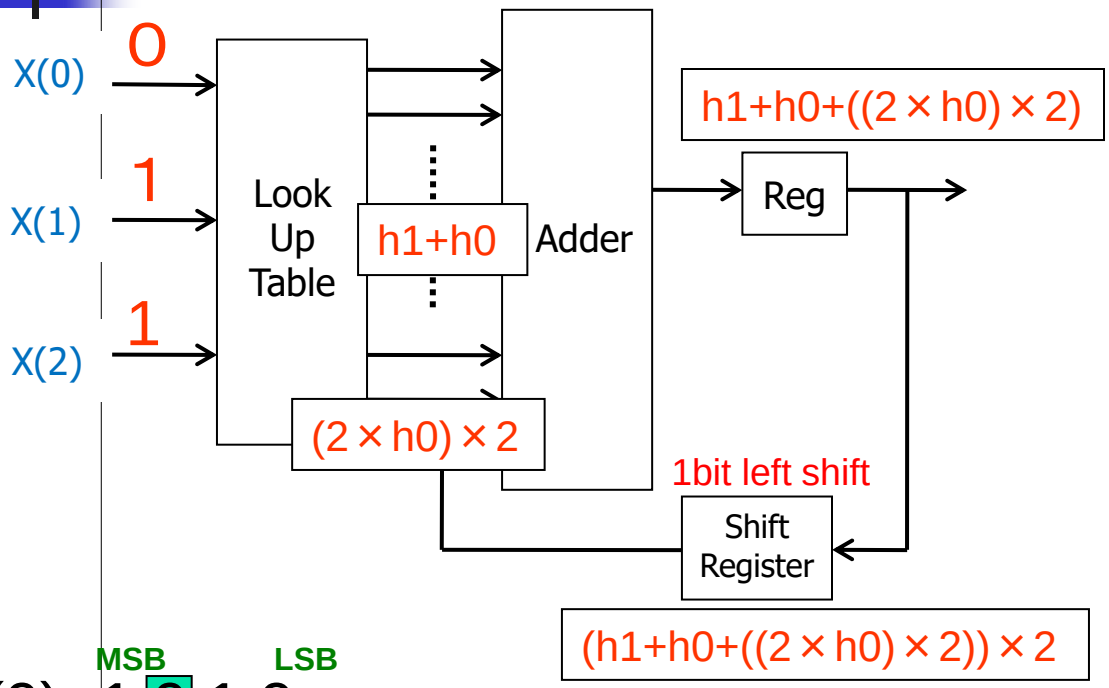
x(0)=1 0 1 0

x(1)=0 1 1 0

x(2)=1 1 0 1

γ	β	α	Y(n)
0	0	0	0
0	0	1	h0
0	1	0	h1
0	1	1	h0+h1
1	0	0	h0
1	0	1	2*h0
1	1	0	h1+h0
1	1	1	2*h0+h1

分散型積和演算　動作原理



γ	β	α	Y(n)
0	0	0	0
0	0	1	h0
0	1	0	h1
0	1	1	h0+h1
1	0	0	h0
1	0	1	2*h0
1	1	0	h1+h0
1	1	1	2*h0+h1

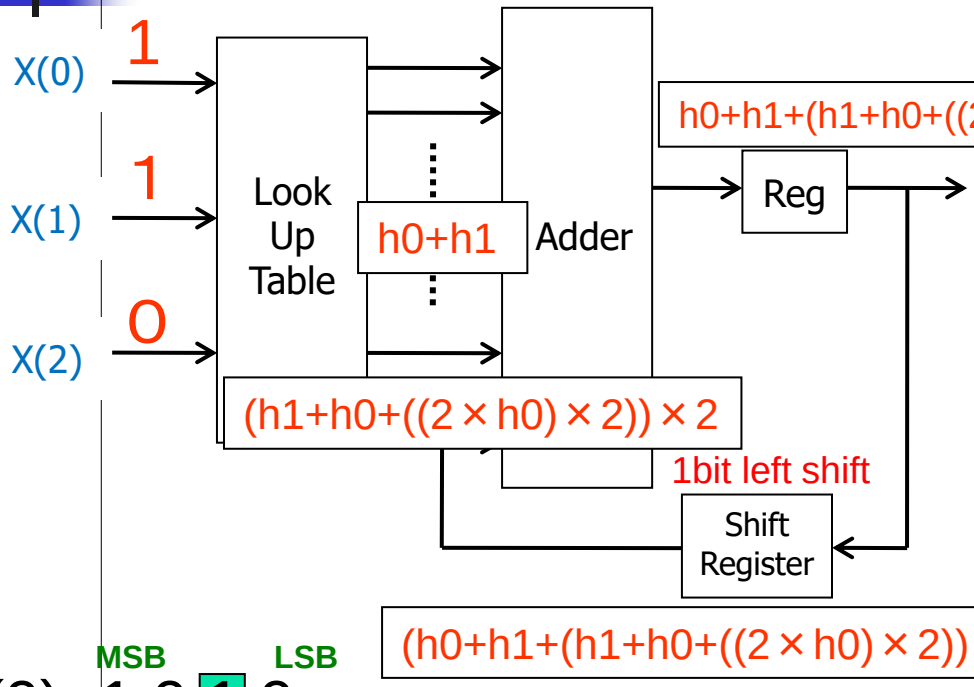
MSB　　LSB

x(0)=1 0 1 0

x(1)=0 1 1 0

x(2)=1 1 0 1

分散型積和演算　動作原理



γ	β	α	Y(n)
0	0	0	0
0	0	1	h0
0	1	0	h1
0	1	1	h0+h1
1	0	0	h0
1	0	1	2*h0
1	1	0	h1+h0
1	1	1	2*h0+h1

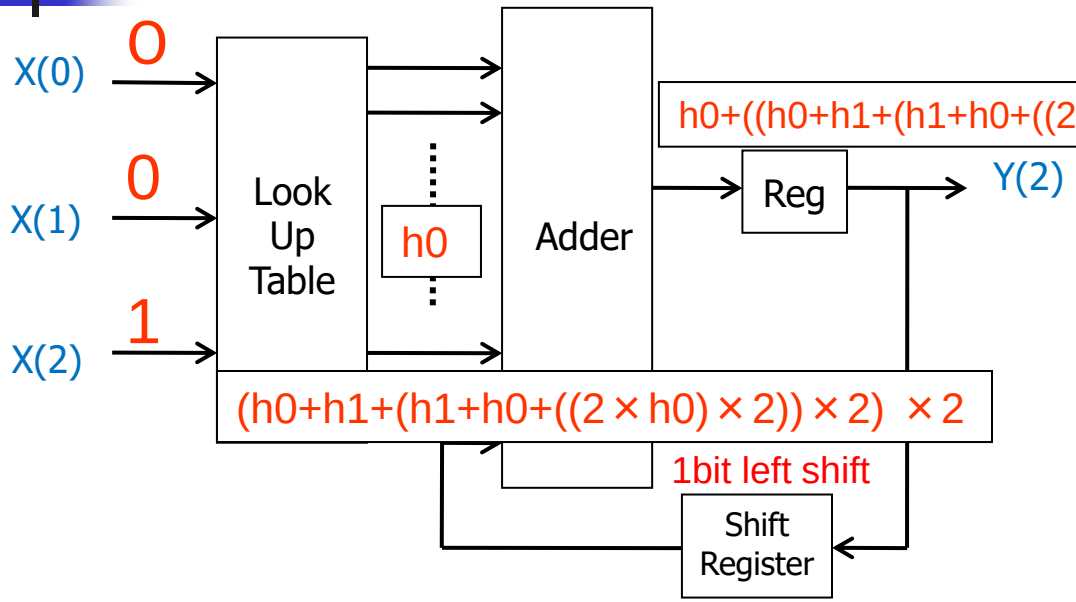
MSB　LSB

x(0)=1 0 1 0

x(1)=0 1 1 0

x(2)=1 1 0 1

分散型積和演算　動作原理



γ	β	α	Y(n)
0	0	0	0
0	0	1	h0
0	1	0	h1
0	1	1	h0+h1
1	0	0	h0
1	0	1	2*h0
1	1	0	h1+h0
1	1	1	2*h0+h1

MSB LSB

x(0)=1 0 1 0

x(1)=0 1 1 0

x(2)=1 1 0 1

分散型積和演算回路のメリット

- 乗算器を使用しない ⇨ 小規模回路・低消費電力
- 積和演算全体の計算時間

  複数の積和演算項を同時にビットシリアル演算

  ⇨ 乗算器を使用する構成と同等

  （積和演算項が多い場合は　同等以上）
- 逐次比較近似ADC　＋　積和演算

  (デジタル制御電源等)

  逐次比較近似ADCは上位ビットから出力される

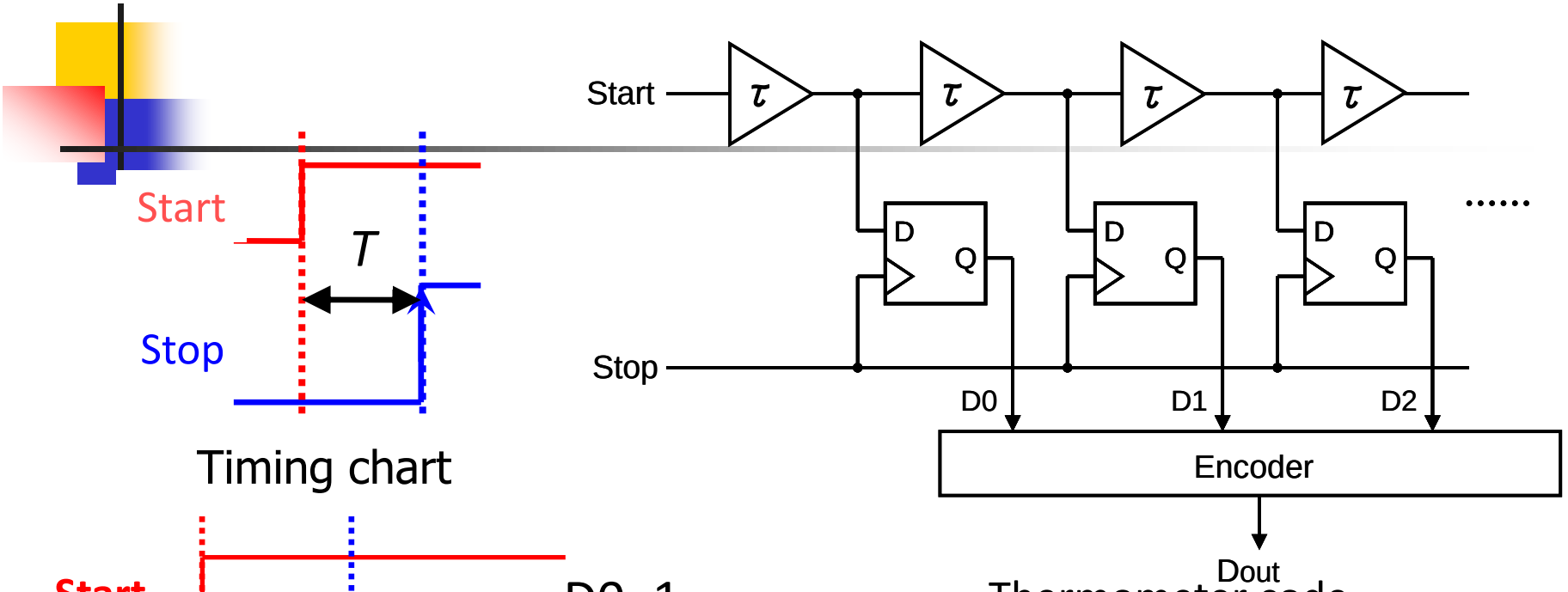
  ⇨ 積和演算をすぐ開始できる （latency が小）
- 分散型積和演算回路単体では MSB からではなく

  LSB から演算することも可能

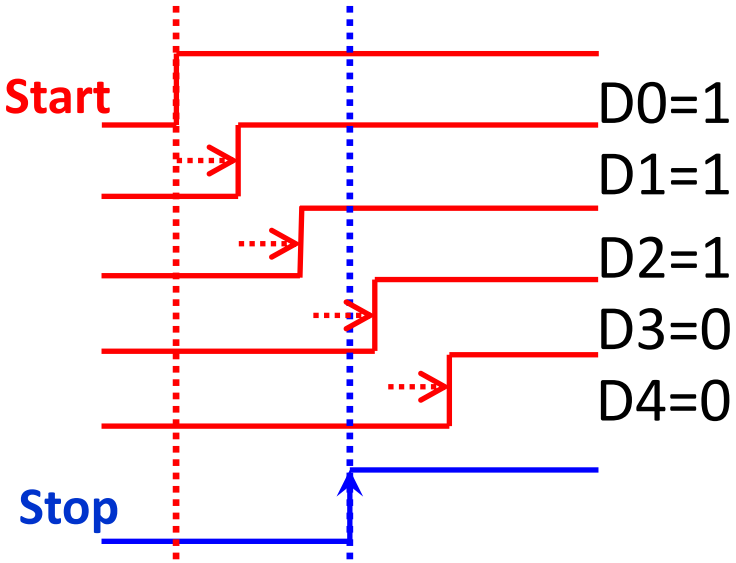
内　容

- セミナーの目的・目標
- デジタル回路の基礎
- スイッチレベル　デジタルCMOS回路
- デジタルCMOS回路の性能（消費電力、スピード）
- 同期回路設計とカウンタ回路
- 加算器、ビットシフト、乗算器
- 事例1：　SAR ADC+分散型積和演算回路
- **事例2：　自己校正機能をもったTDC回路**
- 事例3：　冗長アルゴリズムを用いたSAR ADC
- 最後に

基本TDC (Time-to-Digital Converter) 回路



Timing chart



Thermometer code

⇩

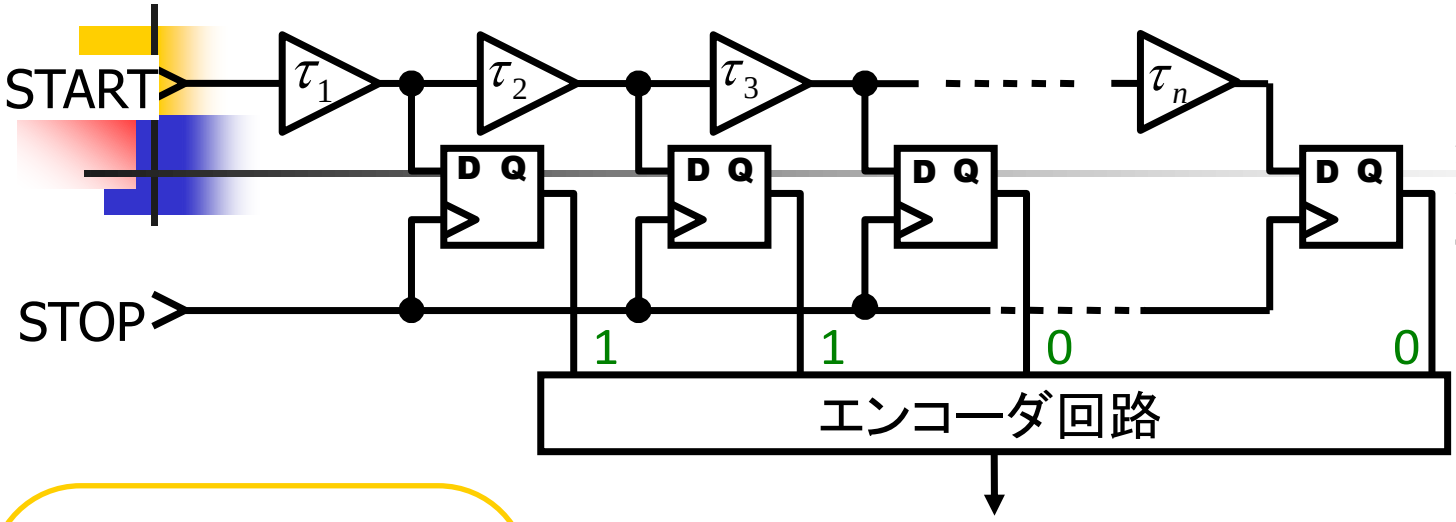
binary code

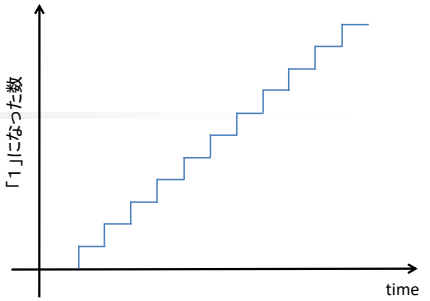
⇩

ディレイタップ何段に相当するかを測定

CMOS微細化とともに高性能化

TDC単調性の問題





TDC単調増加

Dout=2

各DFF出力Dout

0 0 0 0 0 0 0 0	0
1 0 0 0 0 0 0 0	1
1 1 0 0 0 0 0 0	2
1 1 1 0 0 0 0 0	3
1 1 1 1 0 0 0 0	4
1 1 1 1 1 0 0 0	5
1 1 1 1 1 1 0 0	6
1 1 1 1 1 1 1 0	7
1 1 1 1 1 1 1 1	8

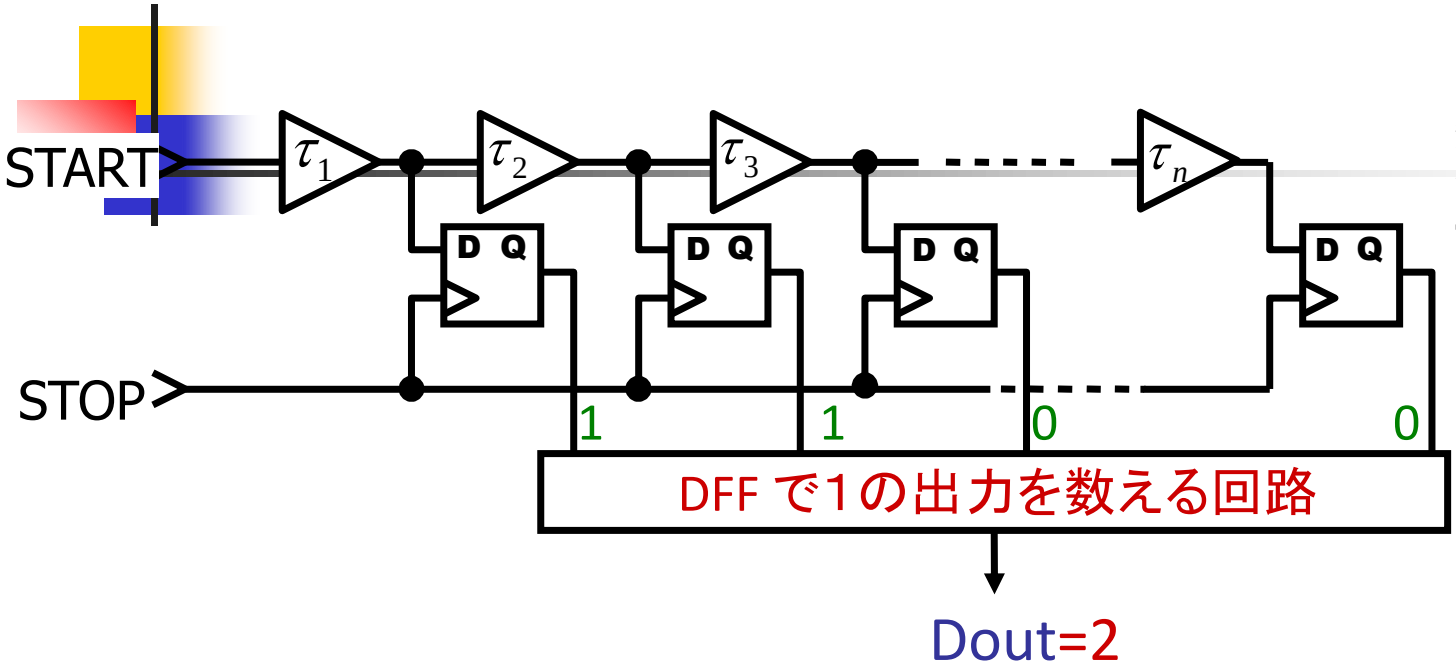
DFF出力のバブルエラー

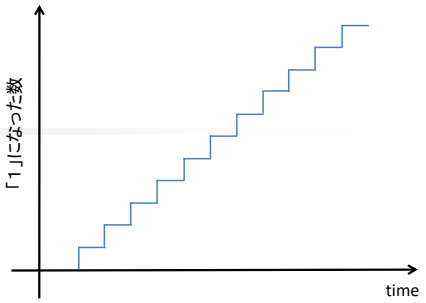
1 0 1 0 0 0 0 0	2
1 1 1 0 0 0 0 0	3
1 1 1 0 1 0 0 0	4
1 1 1 0 1 0 1 0	5
1 1 1 0 1 0 1 1	6

バッファの遅延ばらつき
DFFのオフセットばらつき

バブルエラーが発生

TDC単調性の確保





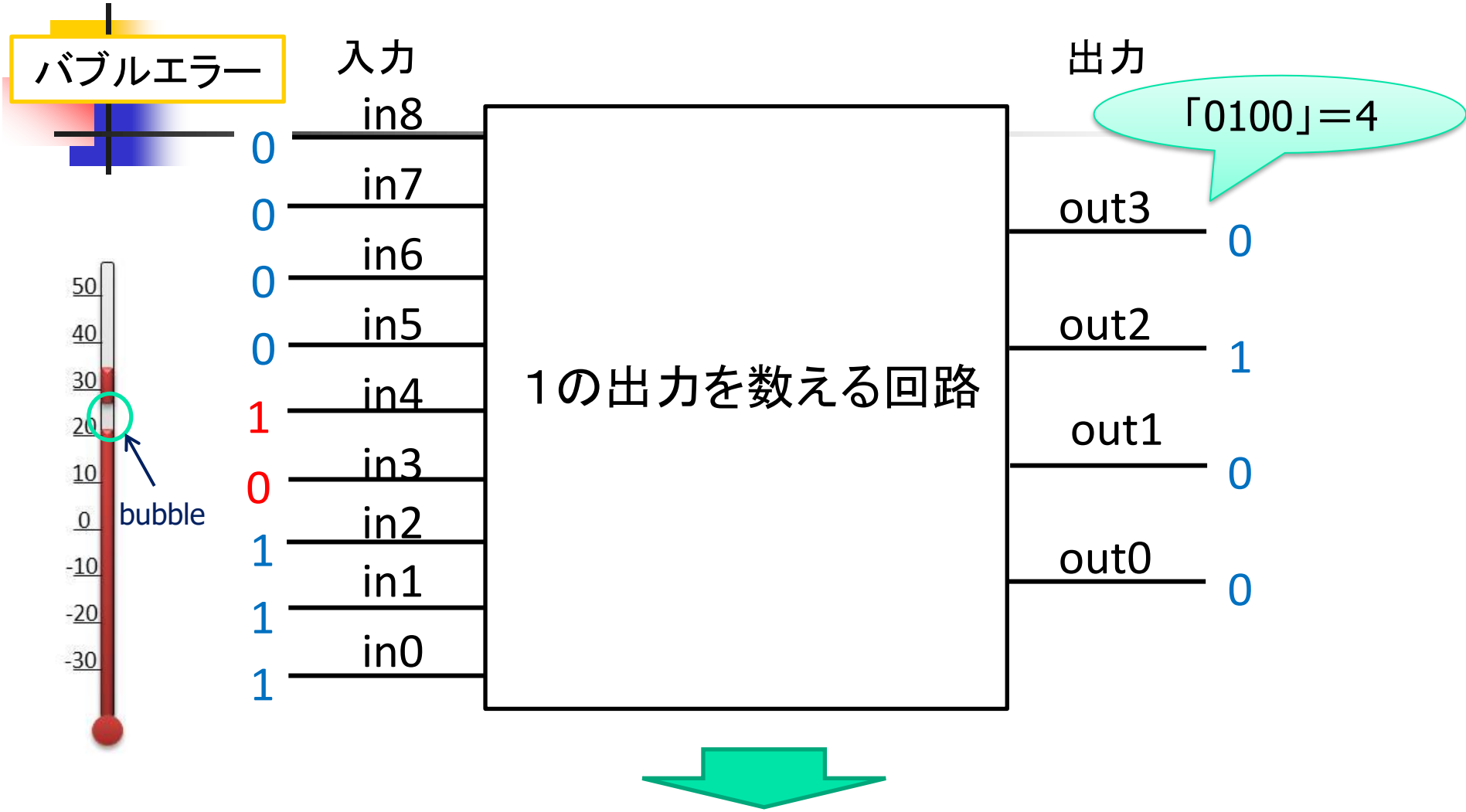
TDC単調増加

時間遅延τ(s)ごとに
DFFの出力「1」の数が
単調に増加

エンコーダ回路をDFFの出力が1のものを数える回路

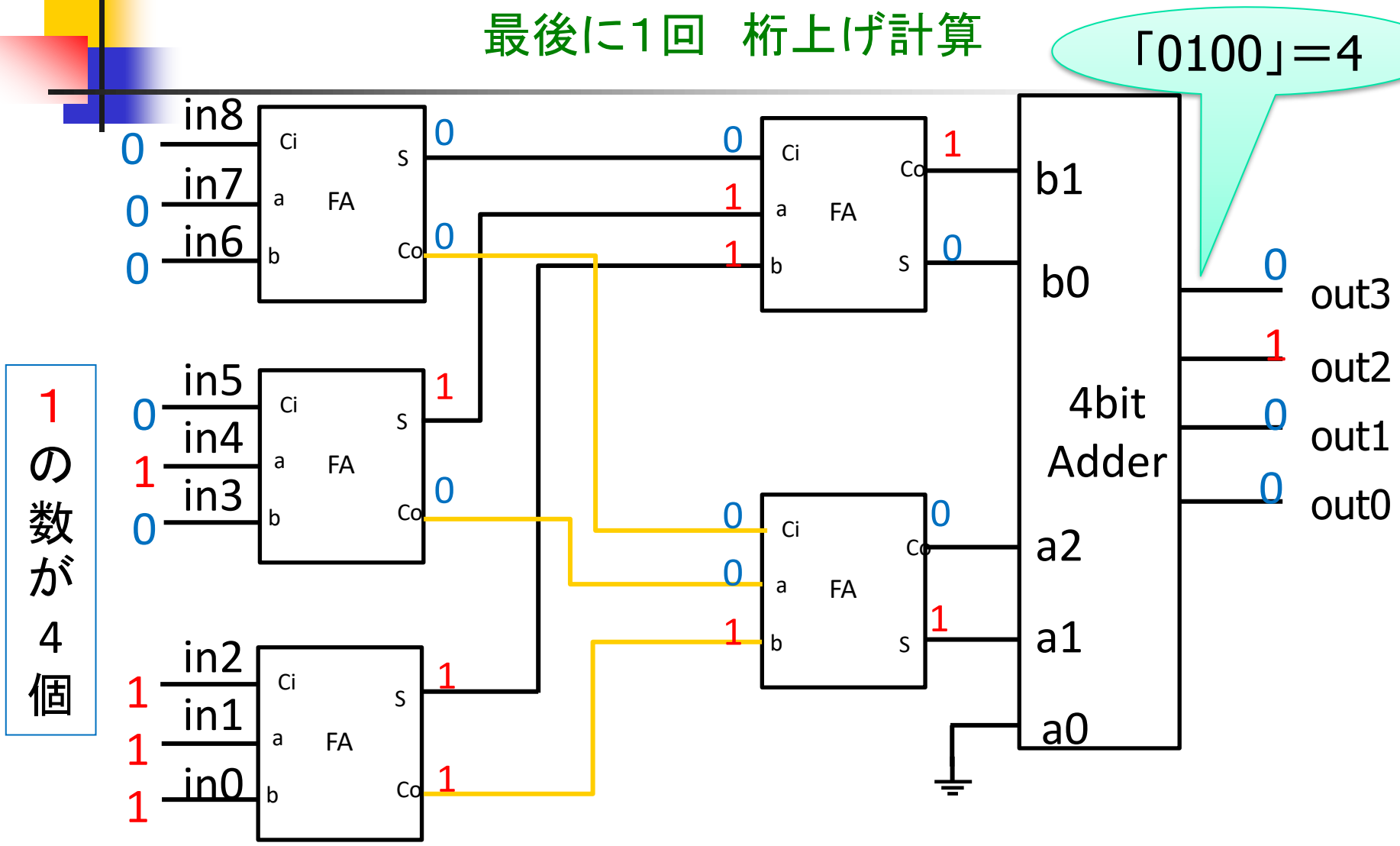
バブルエラーがある場合も単調増加の出力が可能

DFF で1の出力を数える回路



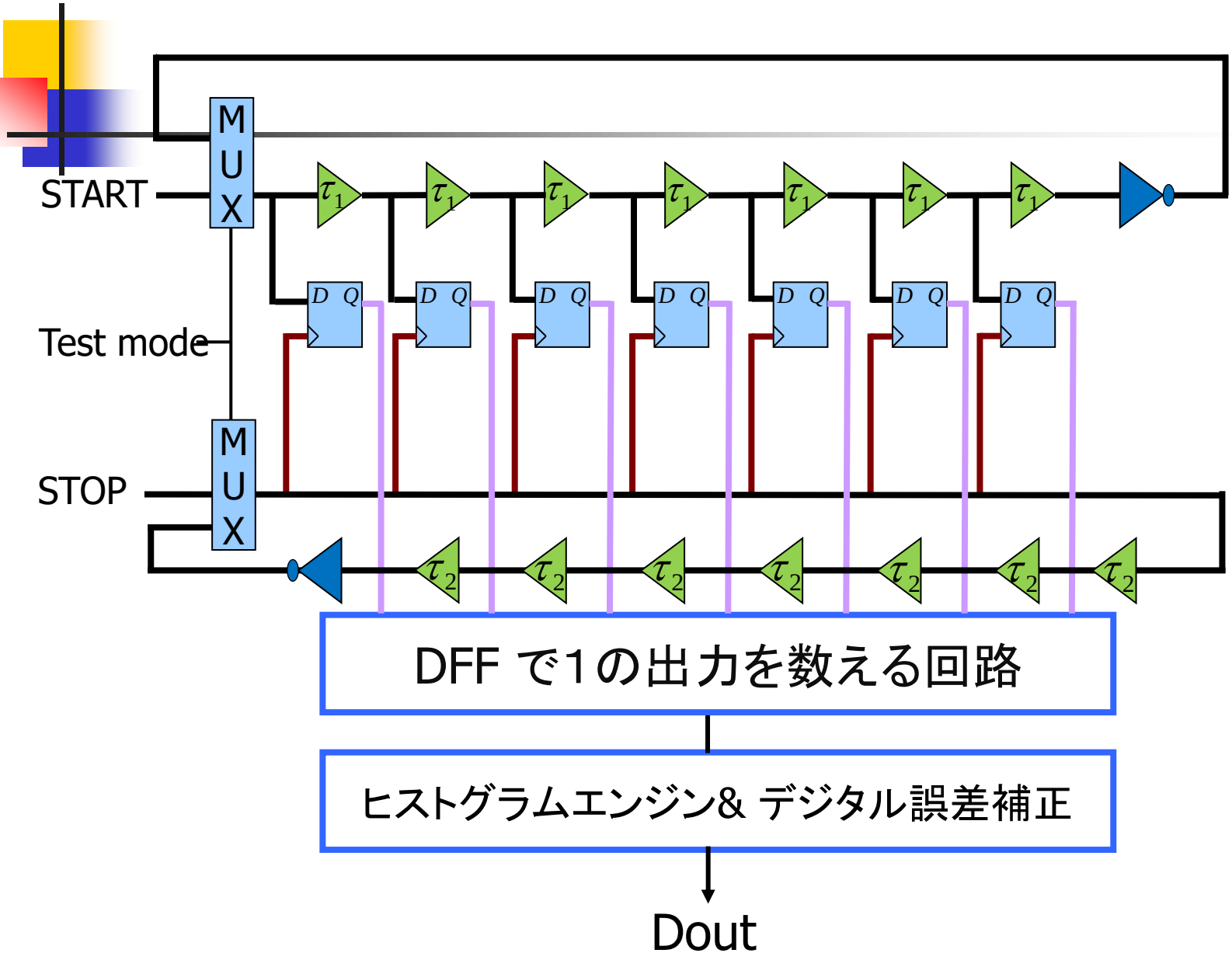
バブルエラーが起きても正しい出力が得られる

DFFの1の出力個数を数える回路

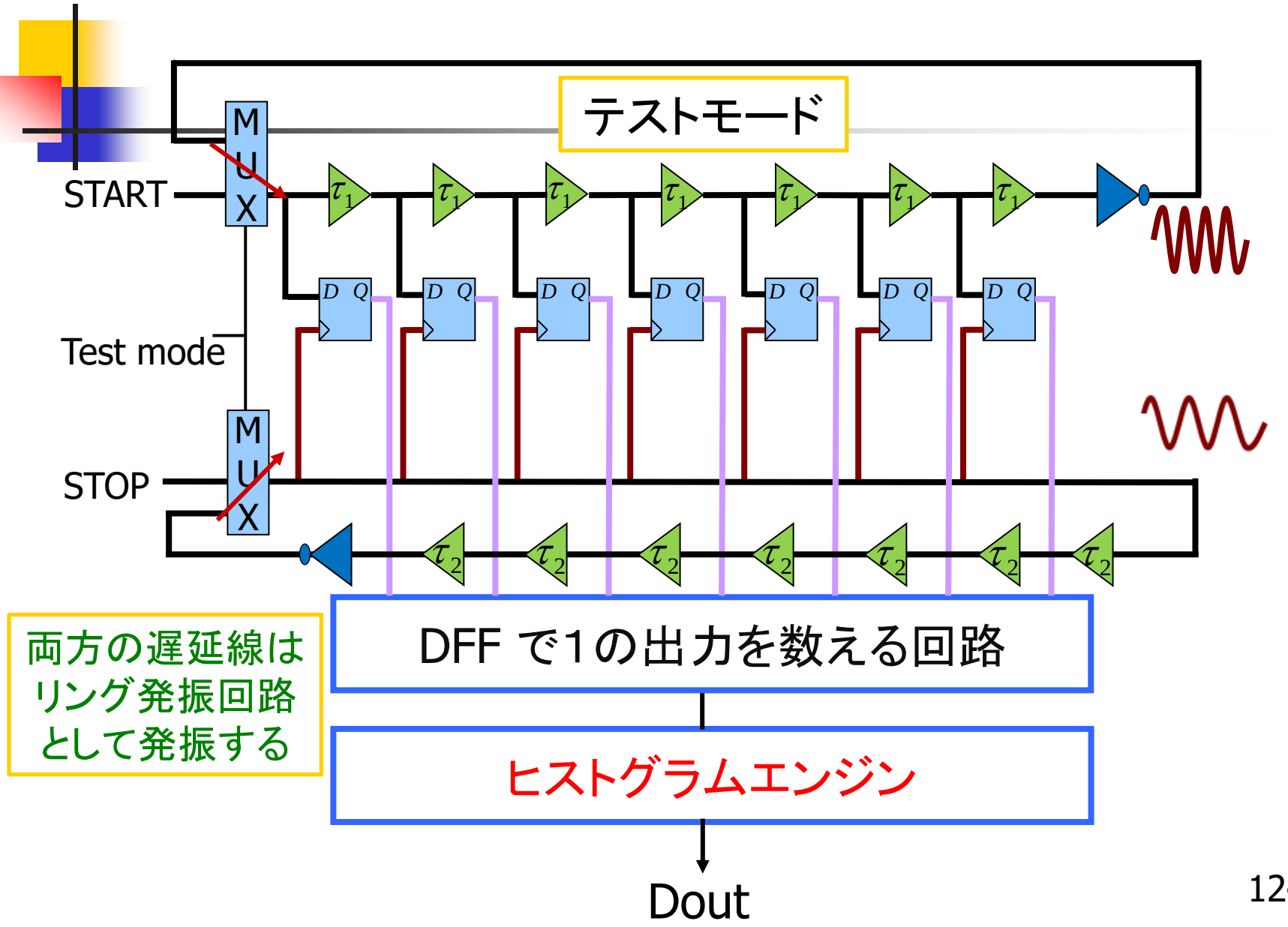


Carry Save Adder

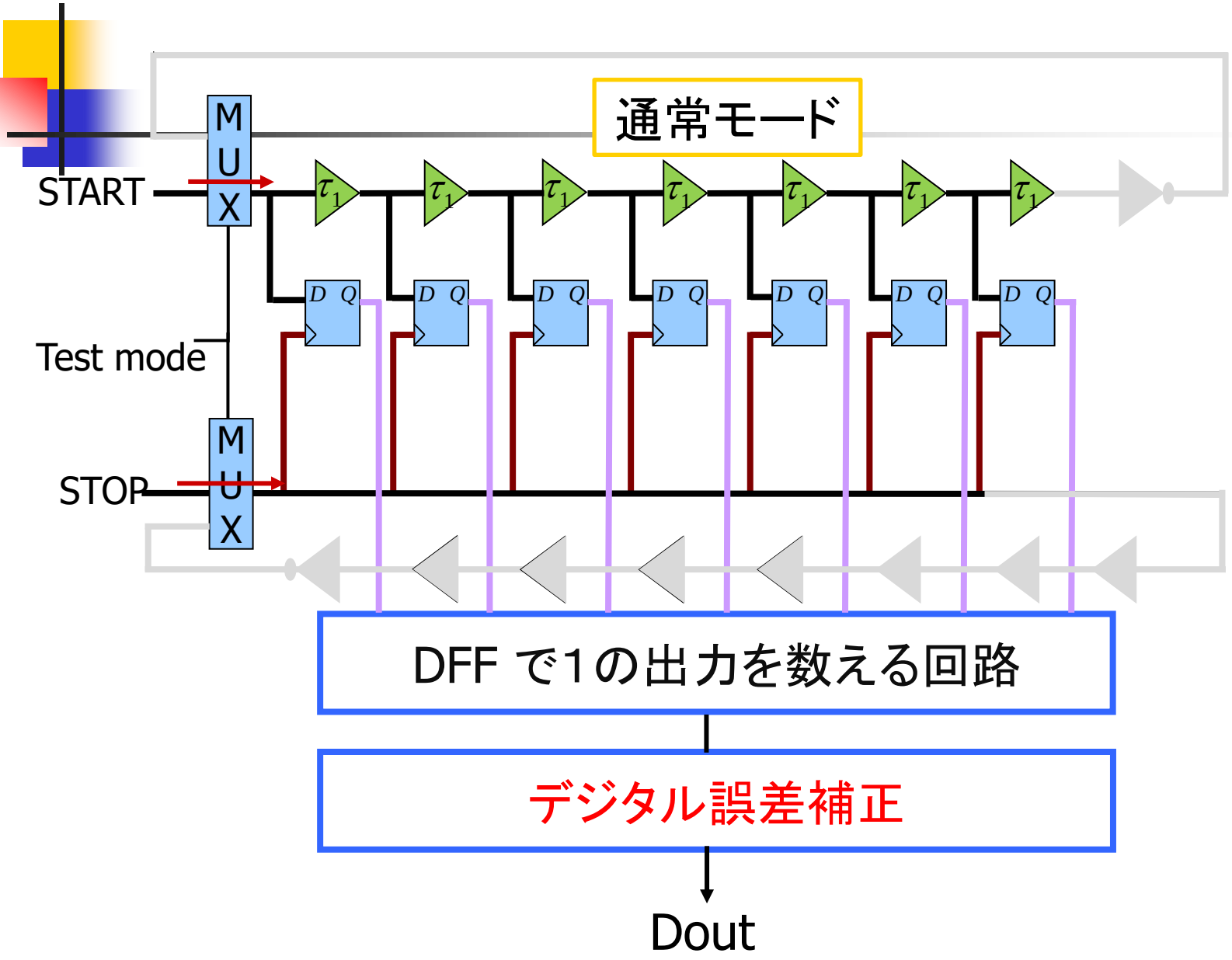
自己校正機能を備えたTDC回路の構成



自己校正機能を備えたTDC回路の構成



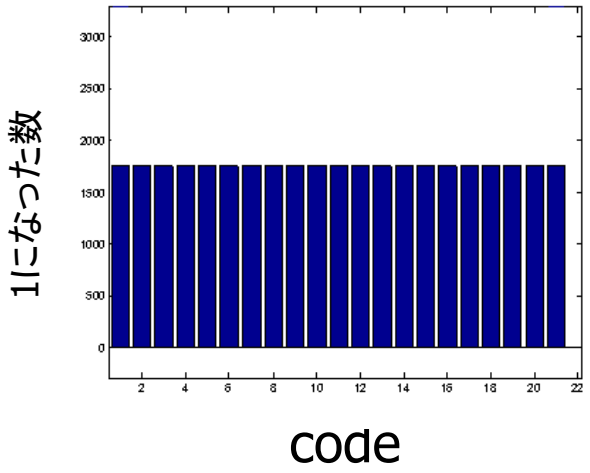
自己校正機能を備えたTDC回路の構成



自己校正の原理 （ヒストグラム法）

テストモード

両方のリング発振器は同期していない（無相関）



TDCが完全に線形

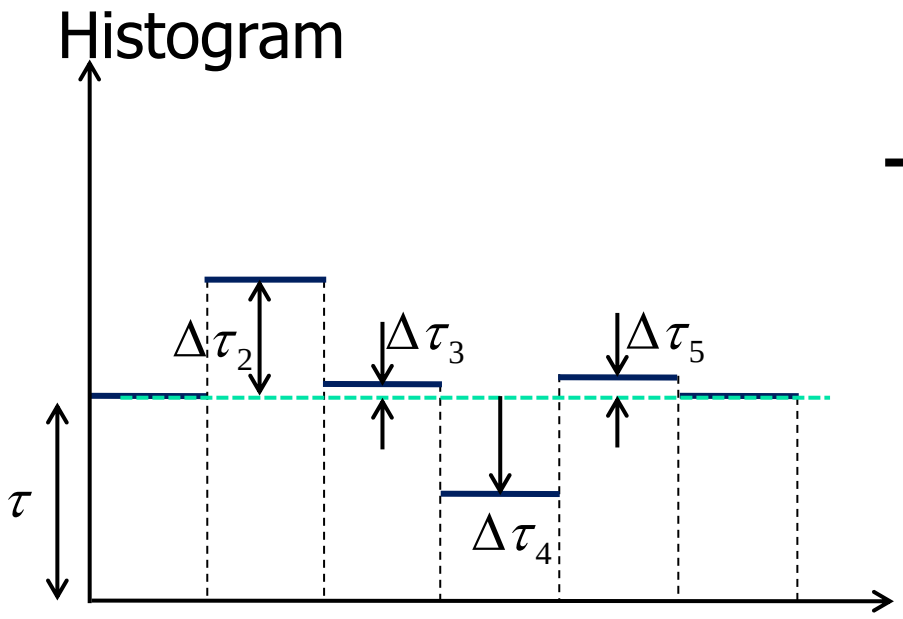
各出現コードの確率が等しい

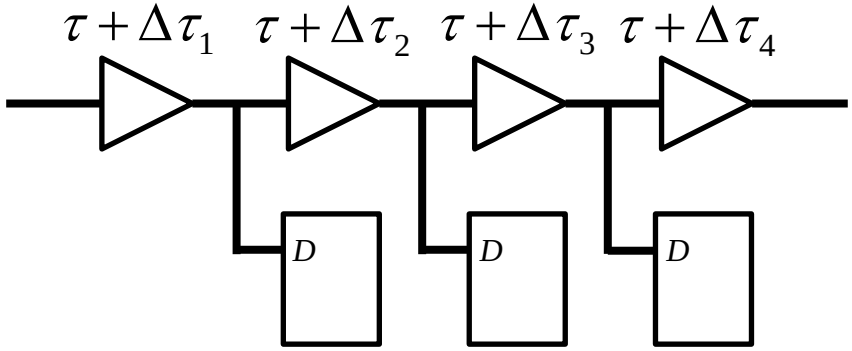
・充分多くの点数をとれば各デジタルコードの
ヒスト グラムは同一になる
・逆に、TDCのヒストグラムデータから
DNL, INL を計算

自己校正の原理（非線形性の同定）

$\tau + \Delta\tau$　　TDCが非線形

TDCの非線形性は遅延ばらつき $\Delta\tau$ によって生じる
INLをヒストグラムより求め逆関数を計算



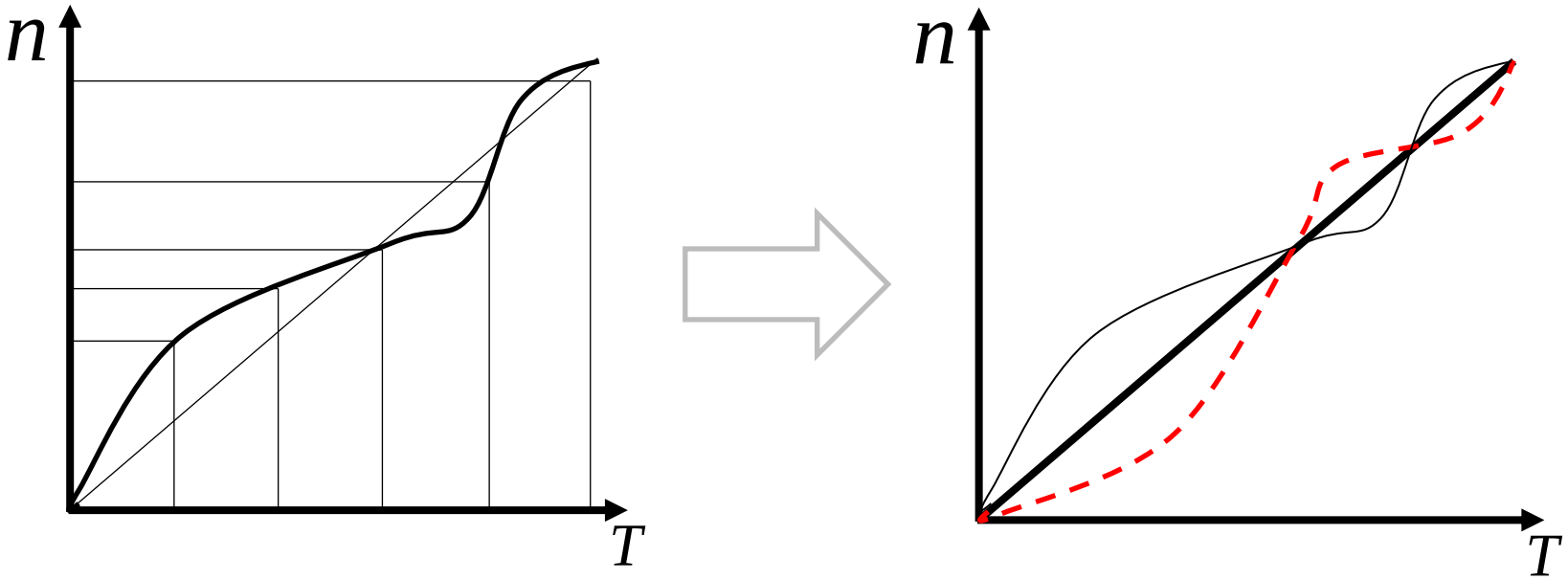


自己校正の原理 (非線形性の補正)

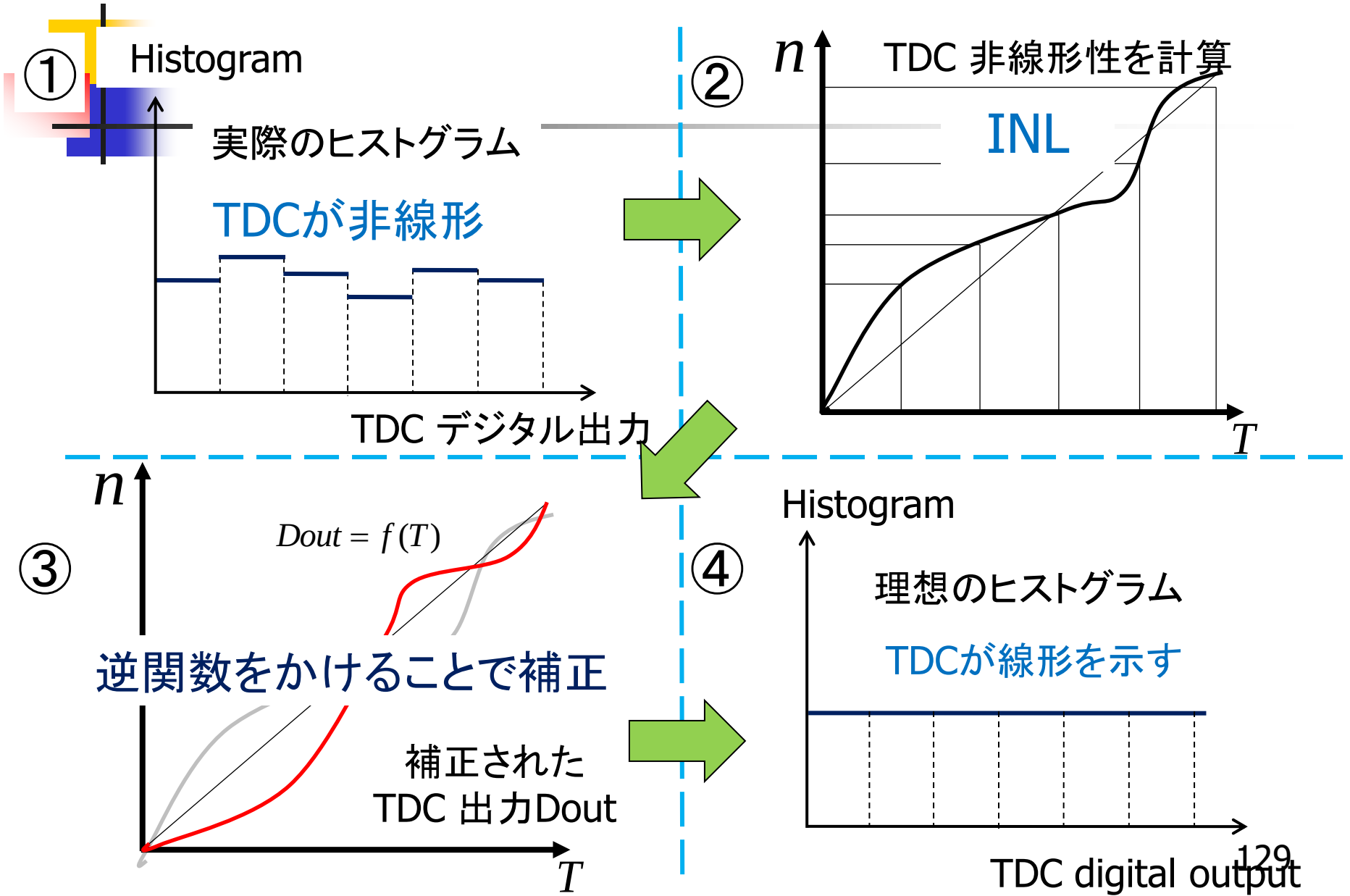
通常モード

非線形性の逆関数をデジタル的にかける

線形性が得られる



非線形性の自己校正

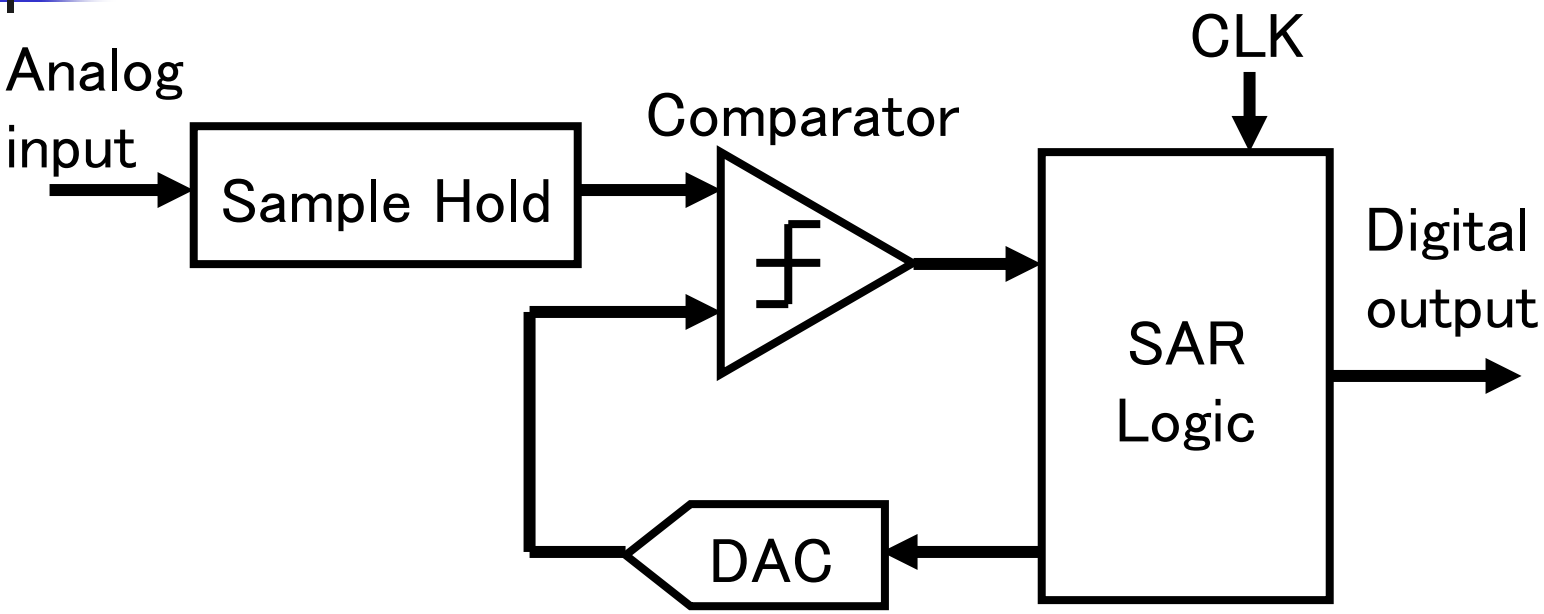

①
Histogram
実際のヒストグラム
TDCが非線形
TDC デジタル出力
②
n
TDC 非線形性を計算
INL
T
③
n
Dout = f(T)
逆関数をかけることで補正
補正された
TDC 出力Dout
T
④
Histogram
理想のヒストグラム
TDCが線形を示す
TDC digital output

内　容

- セミナーの目的・目標
- デジタル回路の基礎
- スイッチレベル　デジタルCMOS回路
- デジタルCMOS回路の性能（消費電力、スピード）
- 同期回路設計とカウンタ回路
- 加算器、ビットシフト、乗算器
- 事例1：　SAR ADC+分散型積和演算回路
- 事例2：　自己校正機能をもったTDC回路
- **事例3：　冗長アルゴリズムを用いたSAR ADC**
- 最後に

逐次比較近似ADC

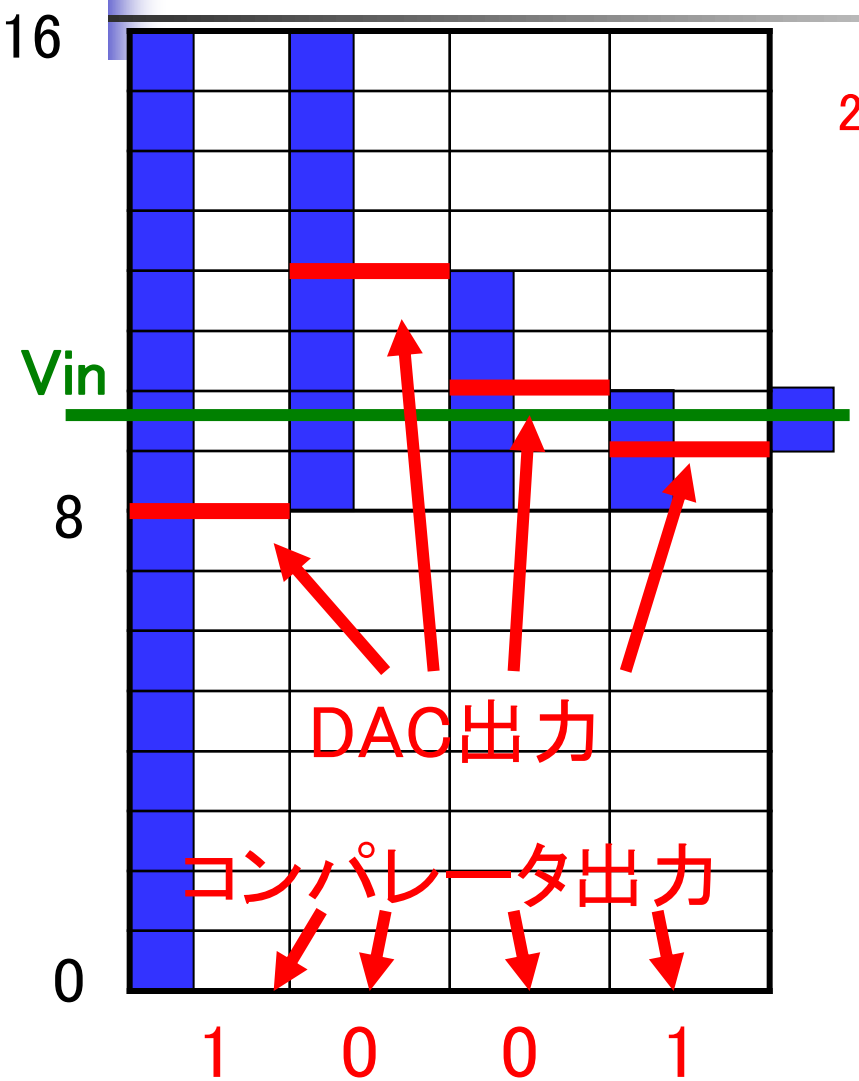
(Successive Approximation Register ADC: SAR ADC)



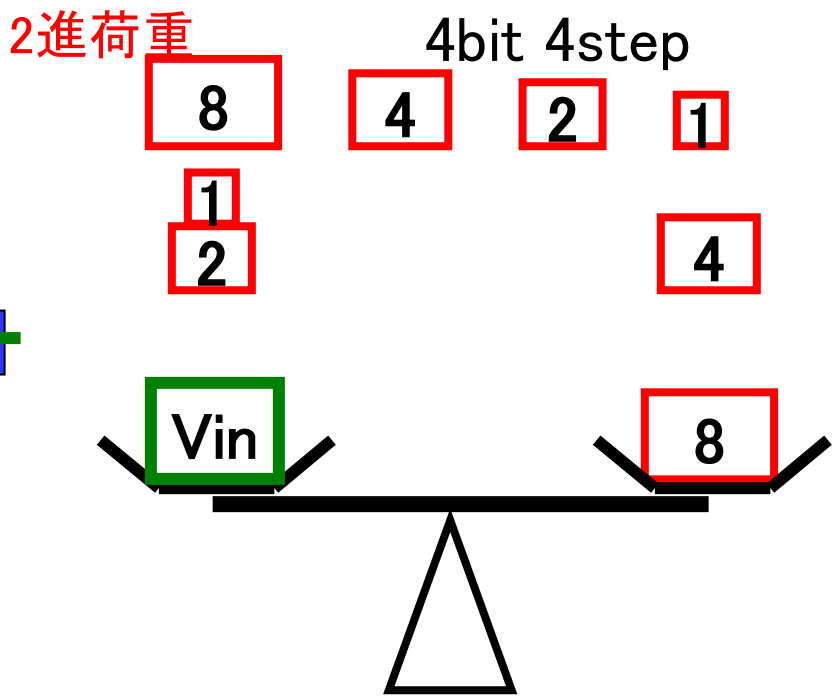
デジタル回路中心，オペアンプ不要.

→ 微細CMOSに適している.

2進探索アルゴリズム

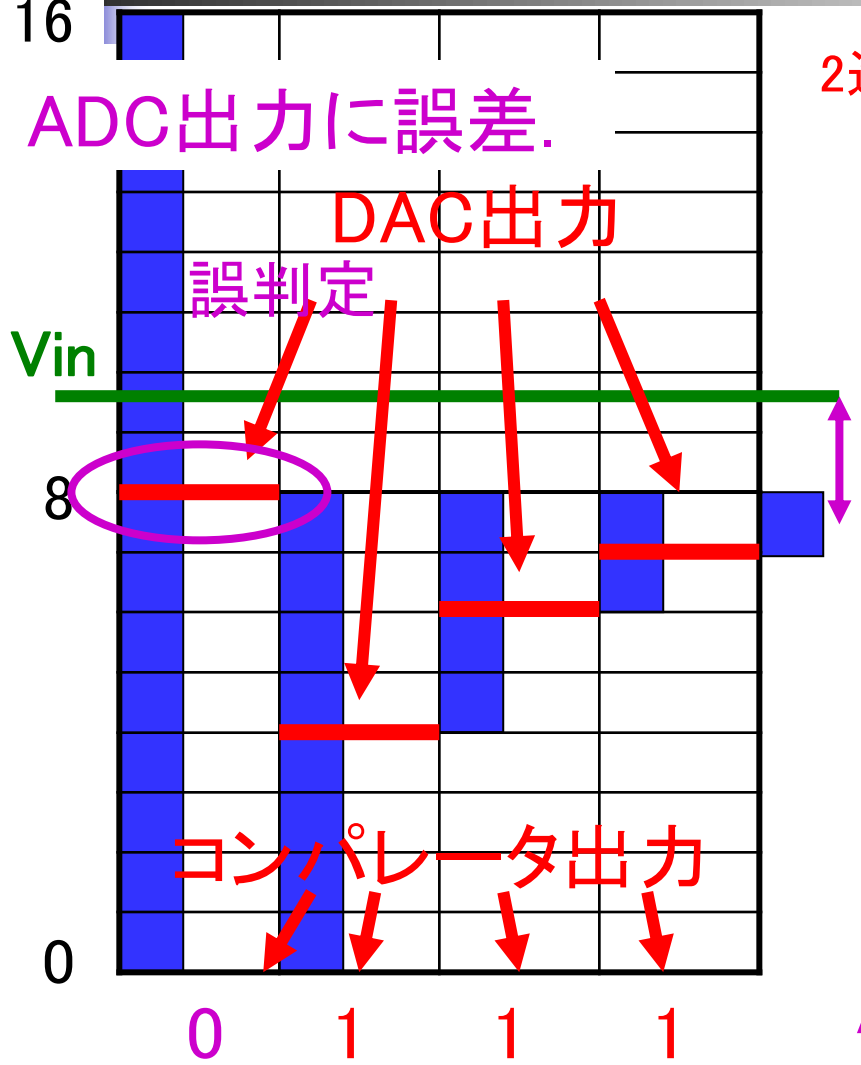


“天秤の原理”

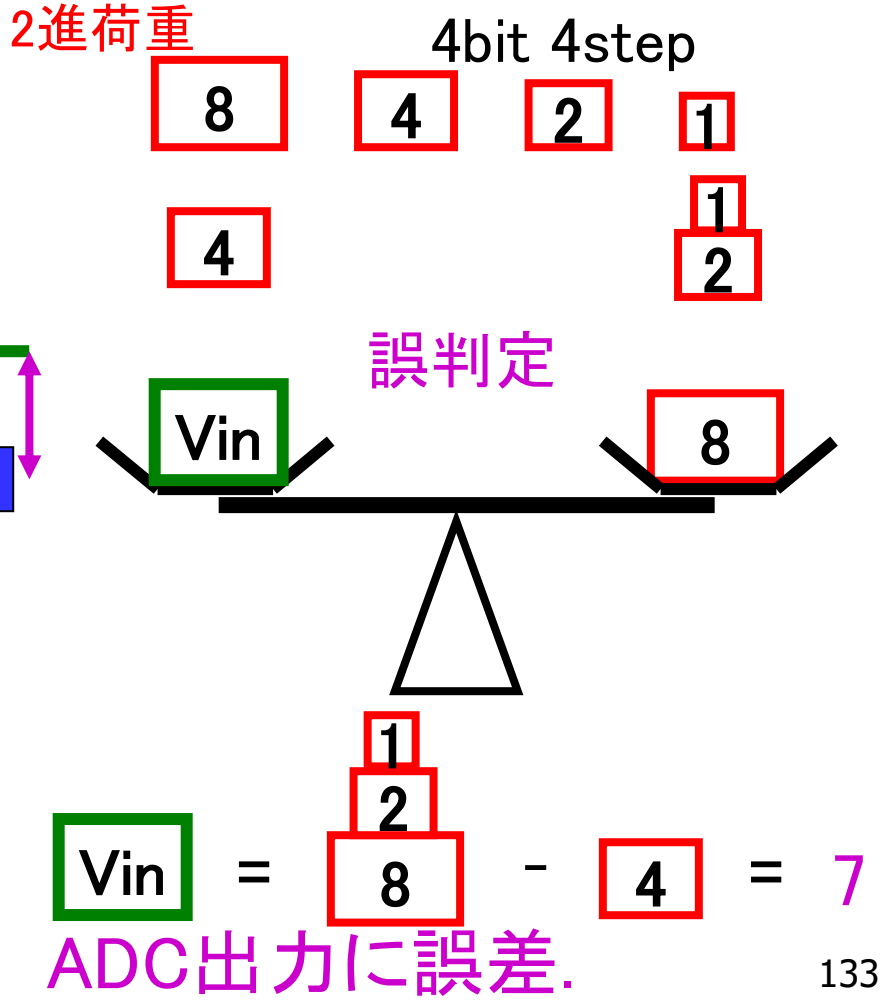


Vin = 4 8 - 1 2 = 9

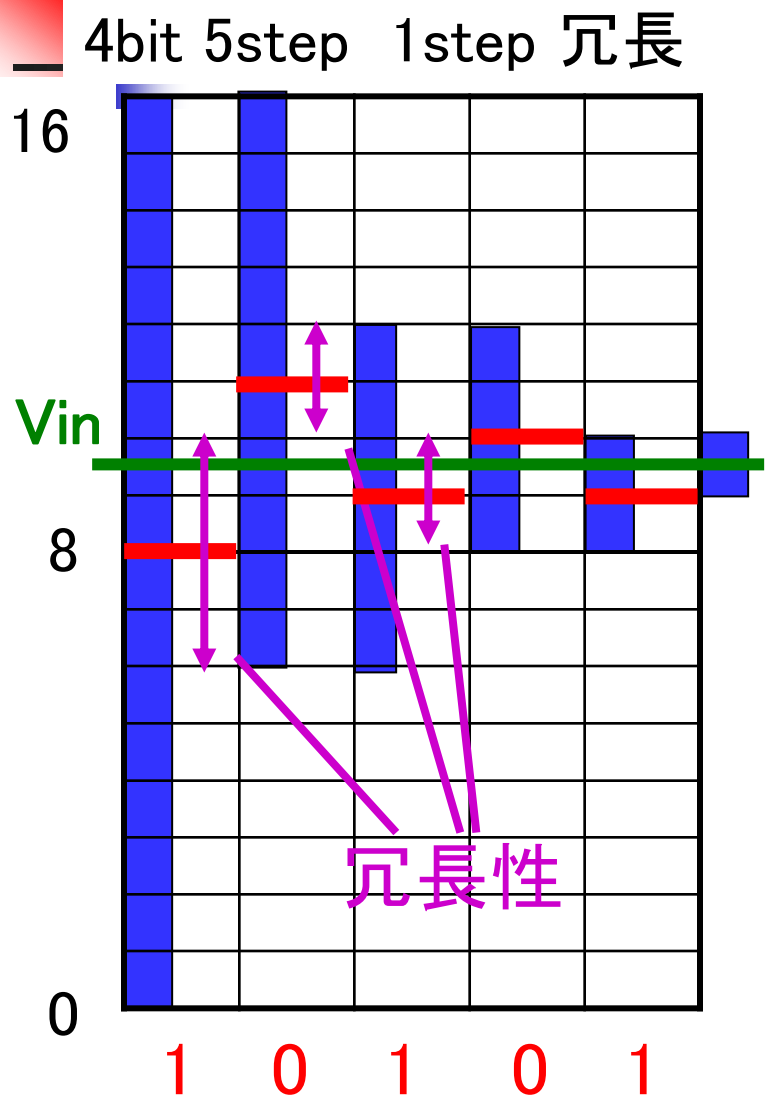
2進探索アルゴリズムの問題点

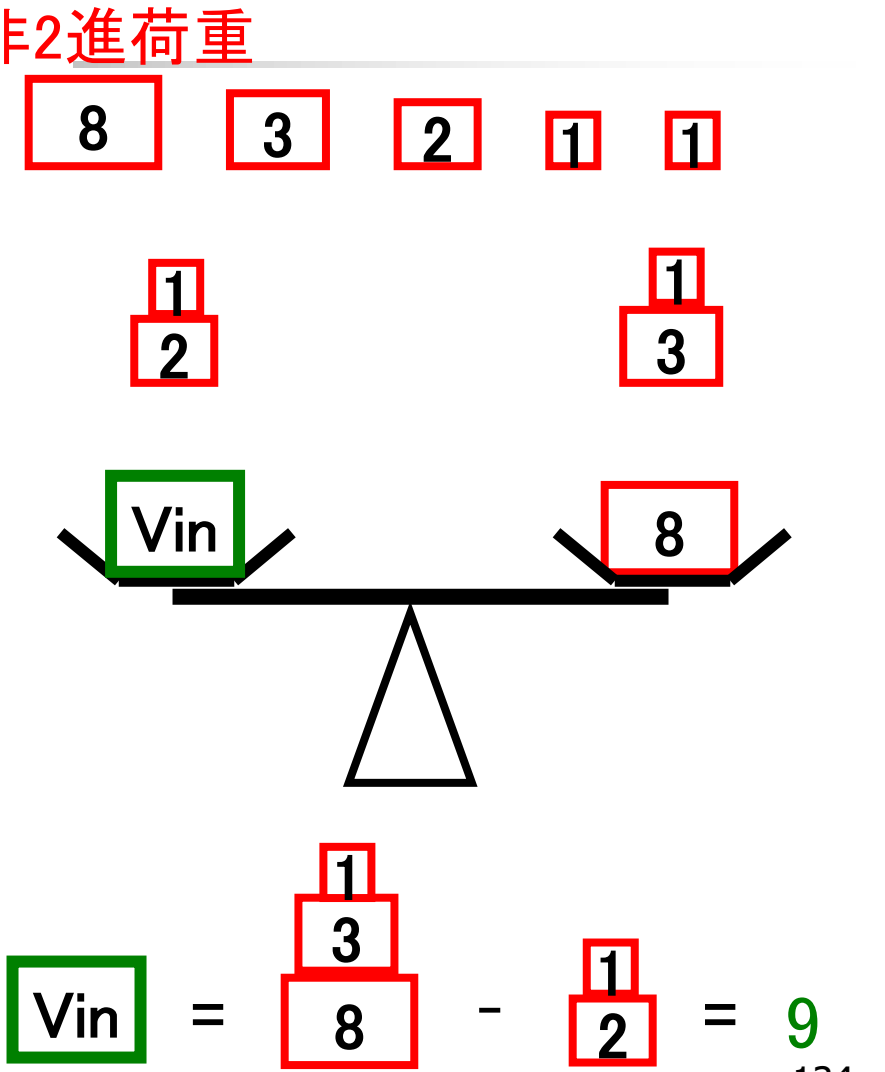


“天秤の原理”

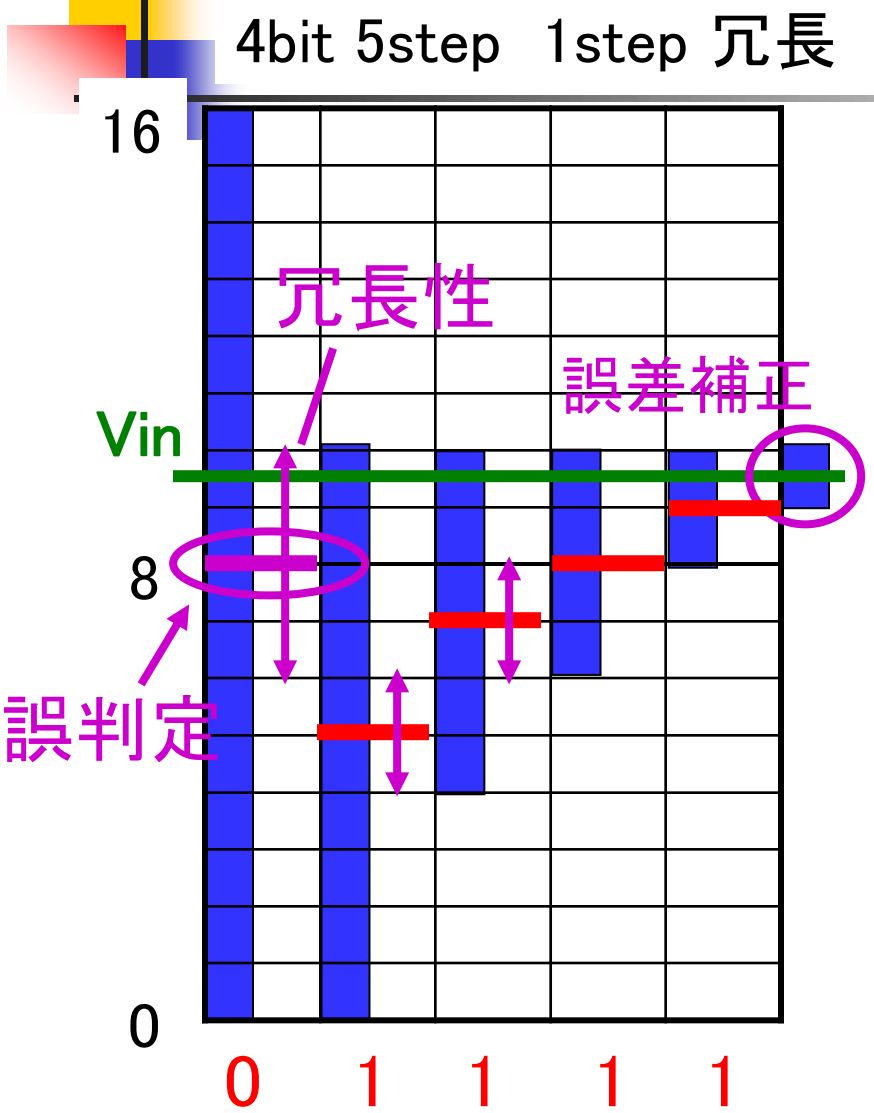


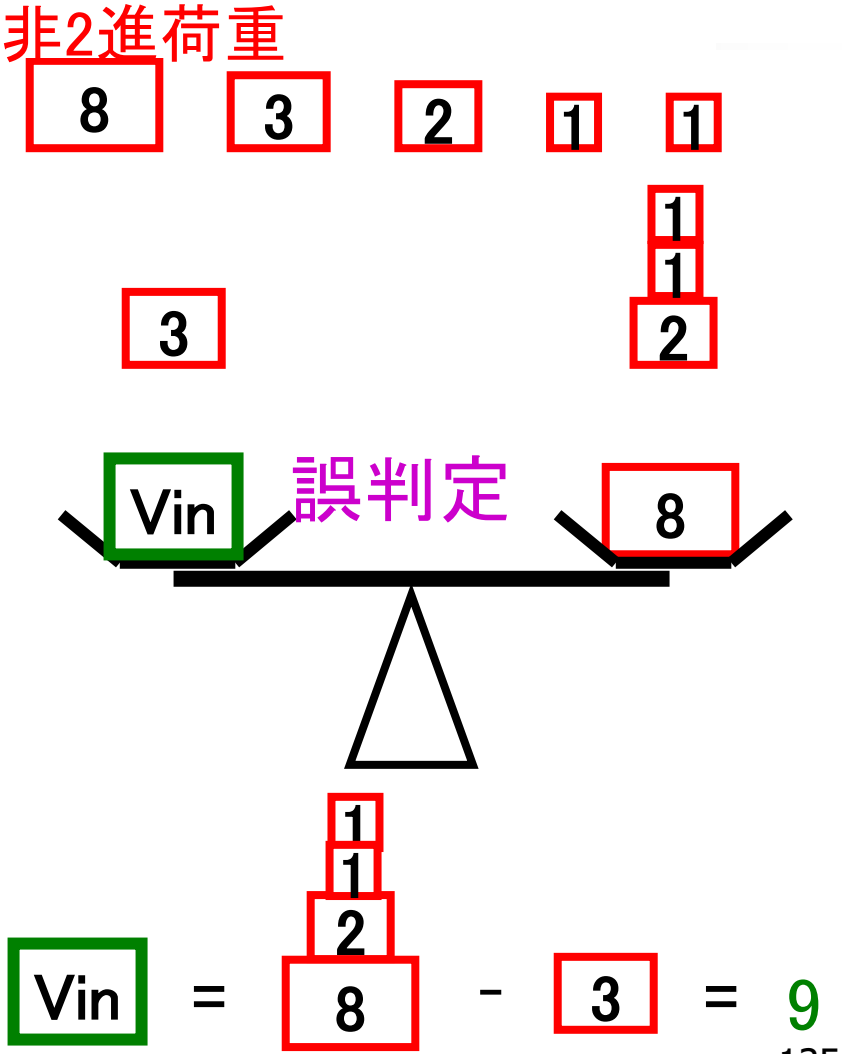
非2進探索アルゴリズム





非2進探索アルゴリズム





誤差補正原理　デジタル出力“9”の場合

2進探索アルゴリズム

誤差補正不可

コンパレータ出力: 1 0 0 1 ← 1パターン

Dout = 8 + 4 − 2 − 1 + 0.5 − 0.5 = 9

非2進探索アルゴリズム

誤差補正可能

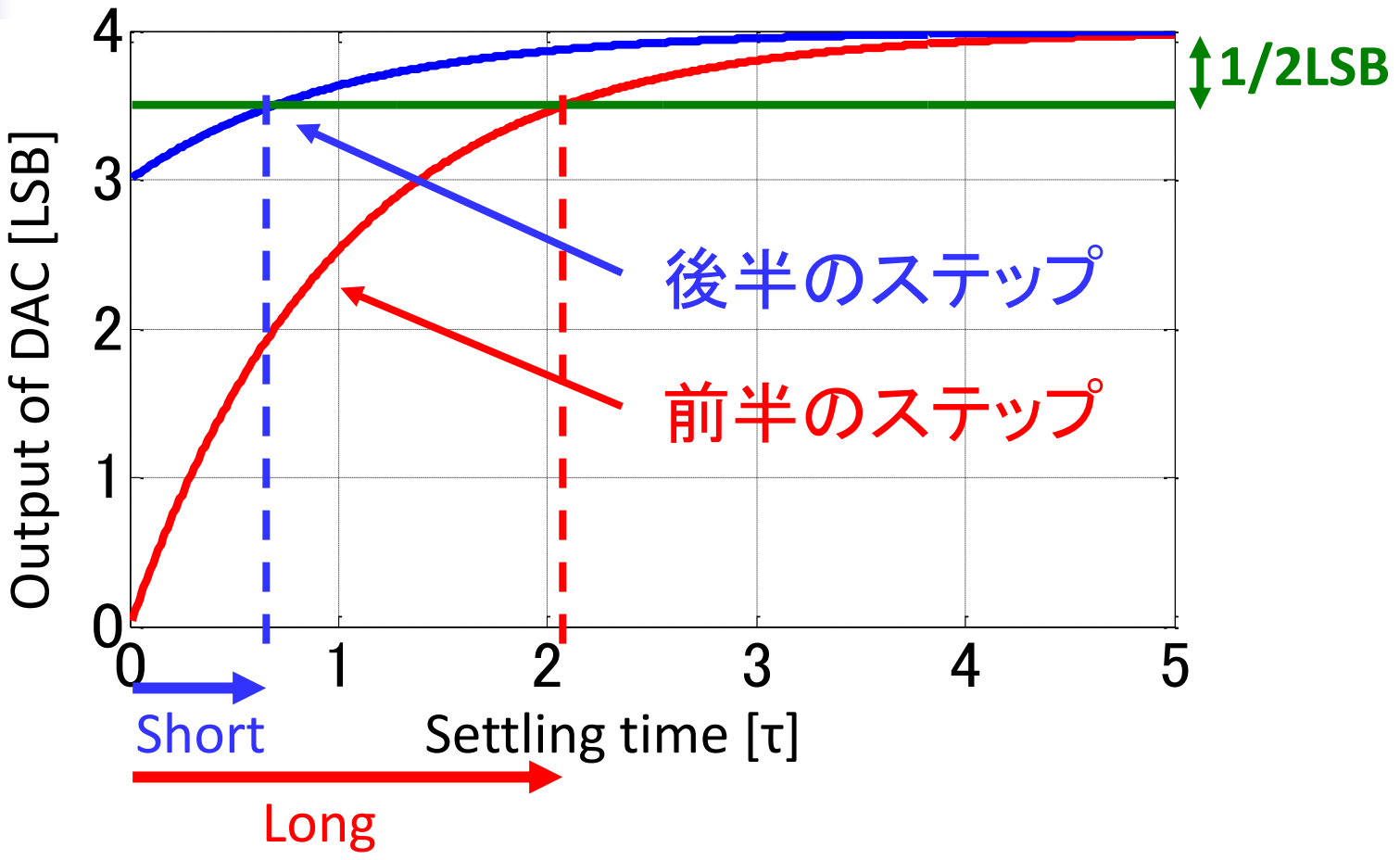
コンパレータ出力: 1 0 1 0 1 ← 複数パターン

Dout = 8 + 3 − 2 + 1 − 1 + 0.5 − 0.5 = 9

コンパレータ出力: 0 1 1 1 1

Dout = 8 − 3 + 2 + 1 + 1 + 0.5 − 0.5 = 9

DAC出力の整定


1/2LSB
後半のステップ
前半のステップ
Output of DAC [LSB]
Settling time [τ]
Short
Long
0
1
2
3
4
5

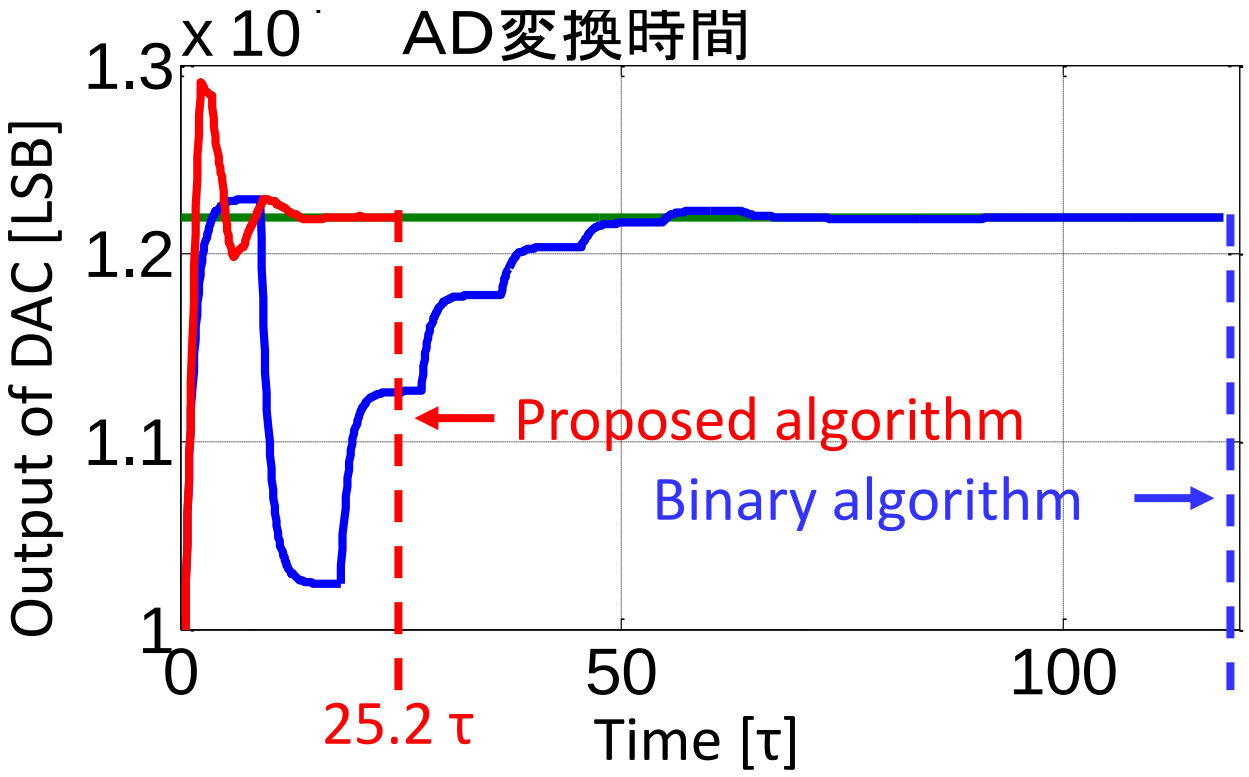
AD変換時間の比較


2進アルゴリズム
4bit
Step1
Step2
Step3
Step4
完全に整定→ 時間：長
AD変換時間
非2進アルゴリズム
Step 1
Step 2
Step 3
Step 4
Step 5
Step 6
不完全整定誤差を補正
不完全整定→ 時間：短

変換時間のMATLABシミュレーション

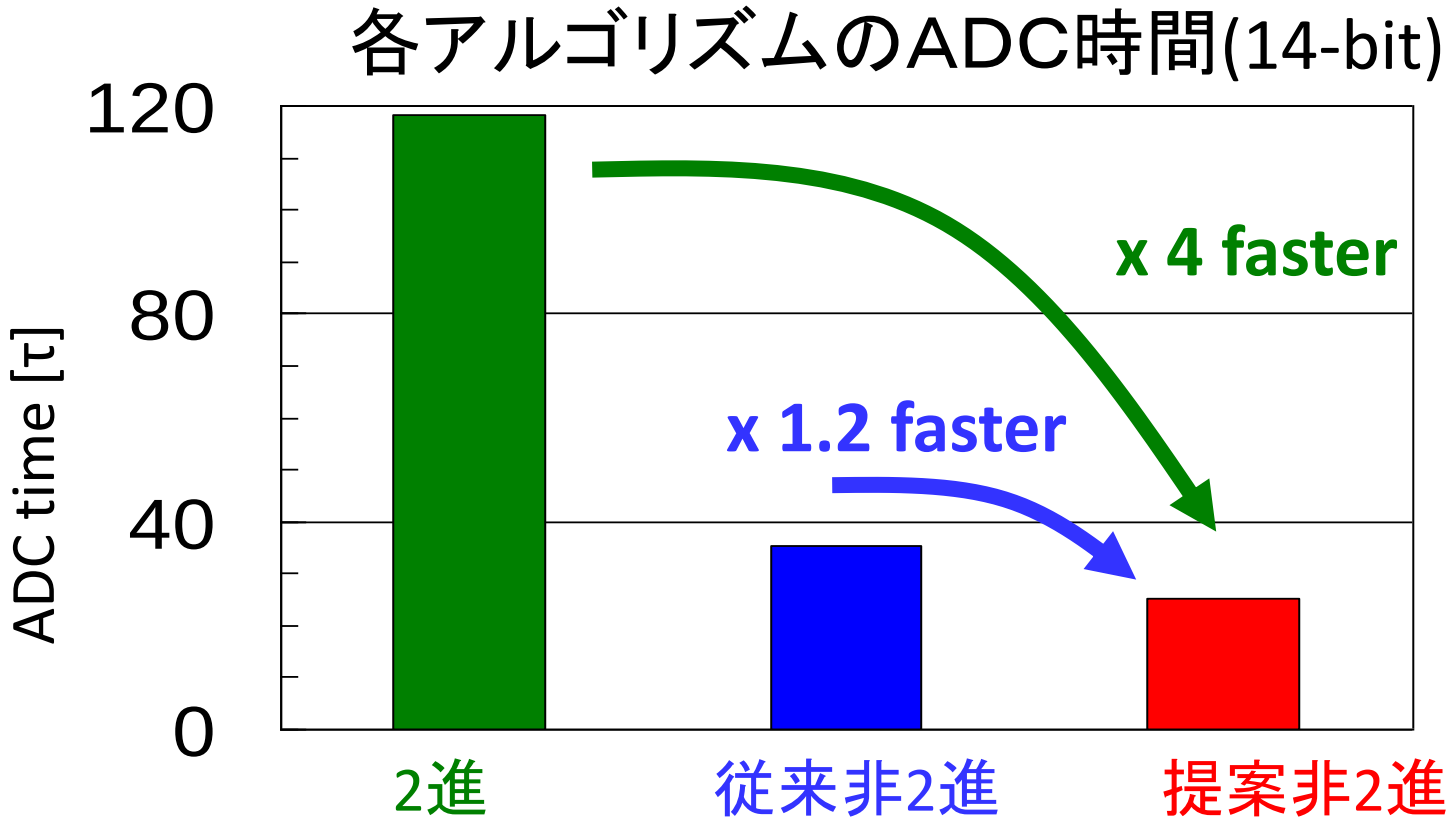
2進アルゴリズム
14-bit,14-step
各ステップでの整定時間: 9.1 τ

提案アルゴリズム
14-bit, 22-step
各ステップでの整定時間: 1.2 τ



整定時間のみを
計算した
シミュレーション

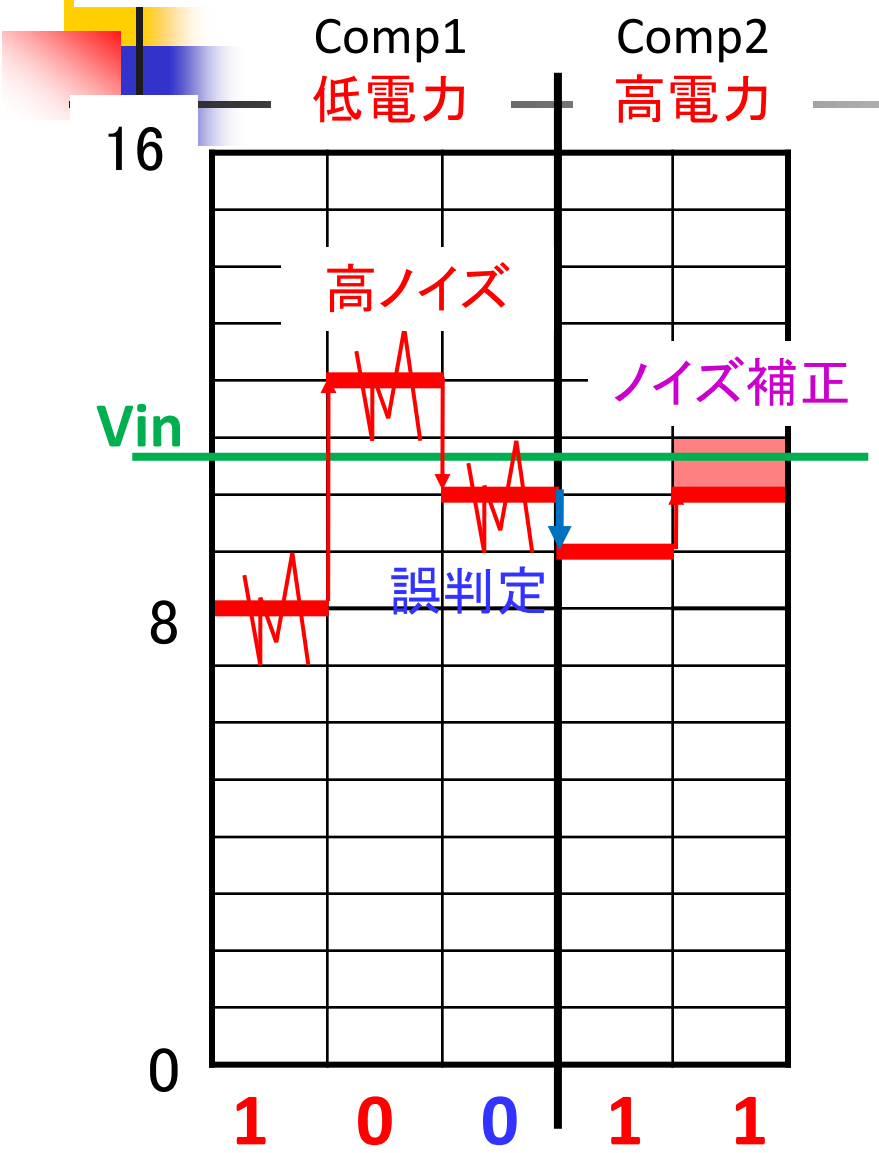
AD変換時間の比較

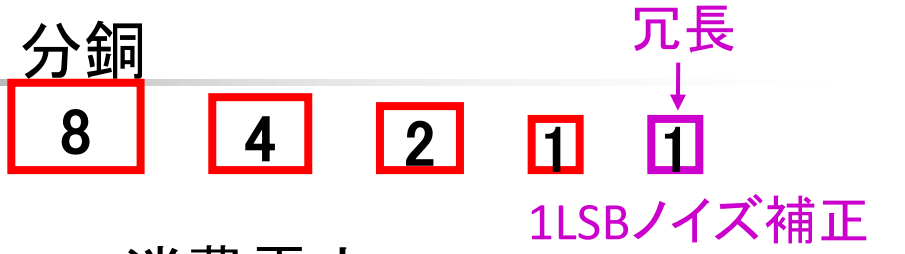


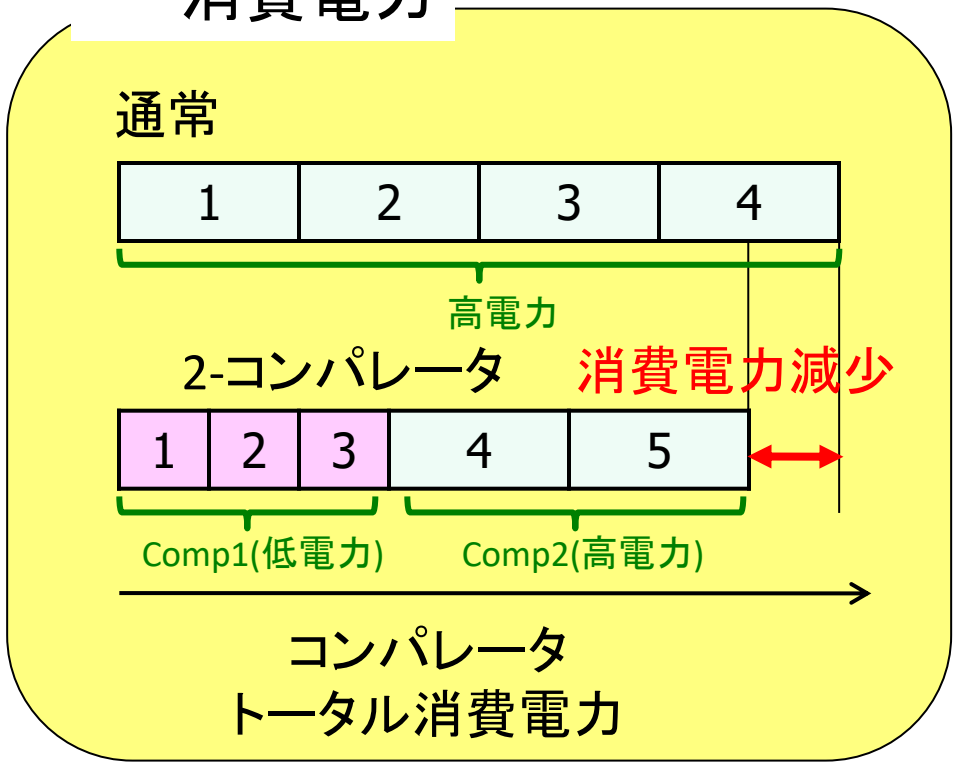
提案アルゴリズム ⟹ ４倍速い

冗長アルゴリズムによる低消費電力化

2つのコンパレータ SAR ADC（IMEC提案）

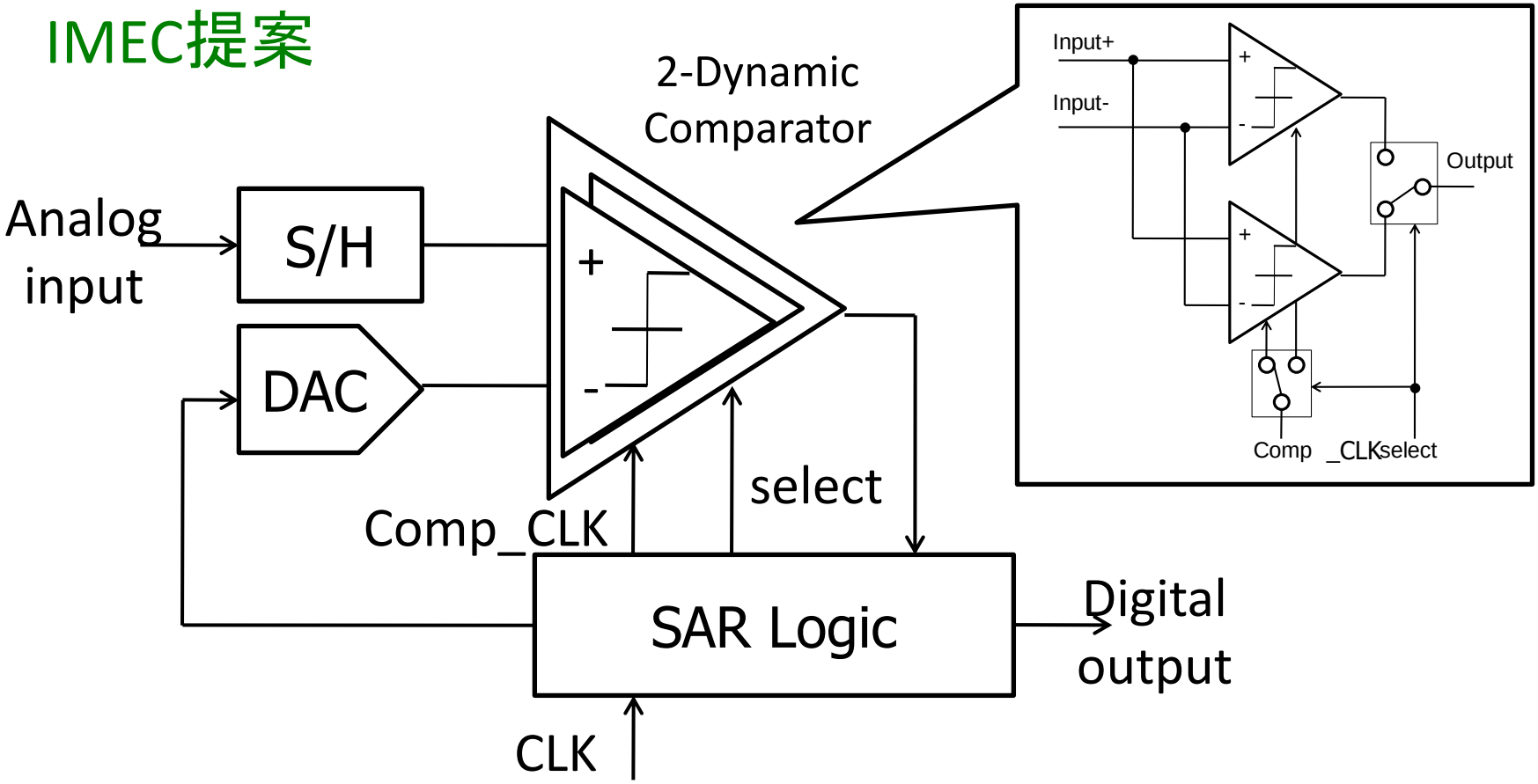






2-コンパレータ SAR ADC構成

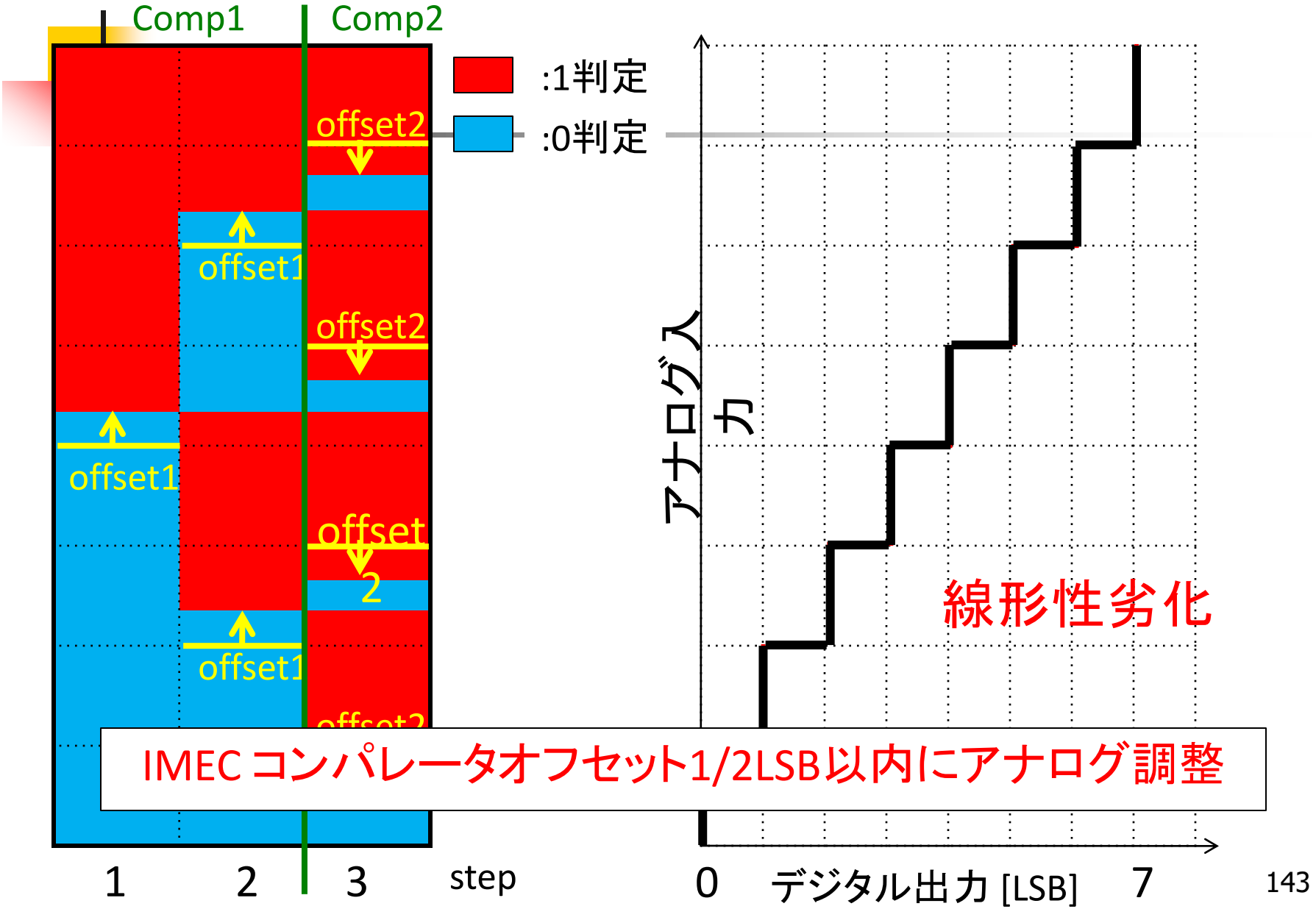
IMEC提案



文献

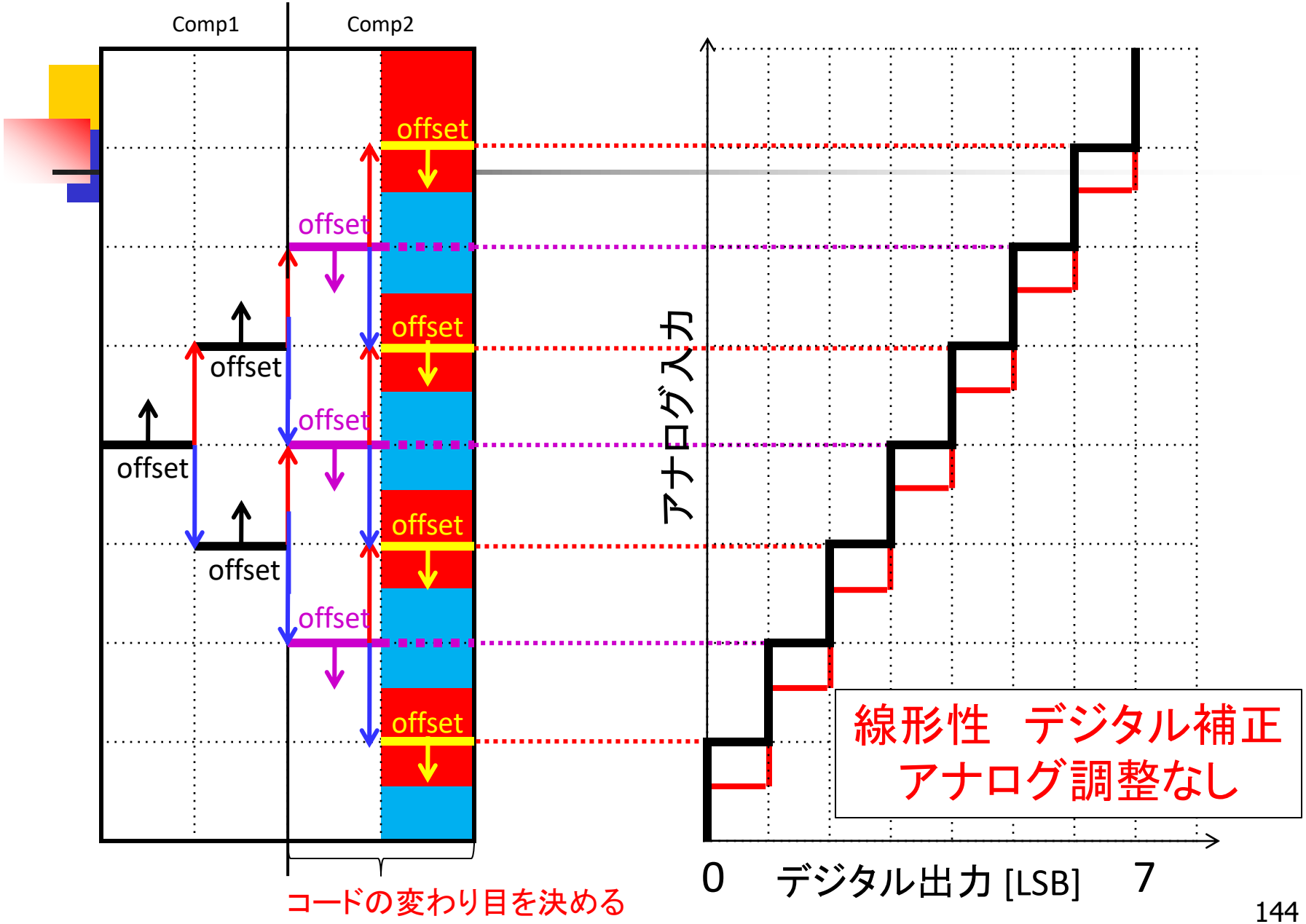
V.Giannini, P.Nuzzo, V.Chironi,A.Baschirotto, G.V.Plas, J.Craninckx
“ An 820 μ W 9b 40MS/s Noise-Tolerant Dynamic-SARADC in 90nm Digital CMOS ” ISSCC (Feb.2008).

2つのコンパレータを用いた技術
コンパレータオフセットミスマッチの影響



IMEC コンパレータオフセット1/2LSB以内にアナログ調整

提案方式　冗長アルゴリズムによるデジタル補正

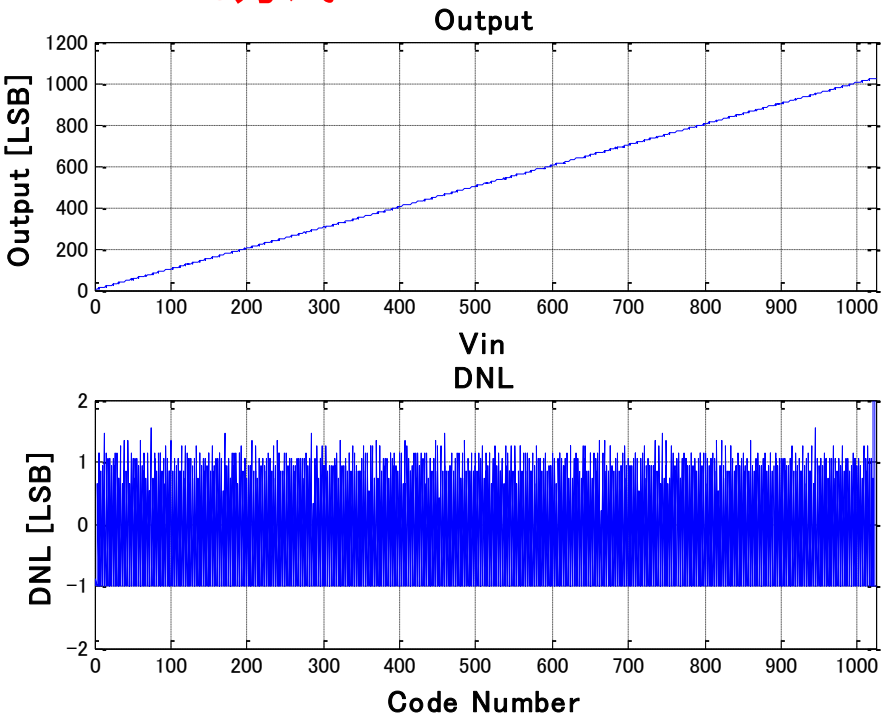


MATLABシミュレーション（ランプ波）

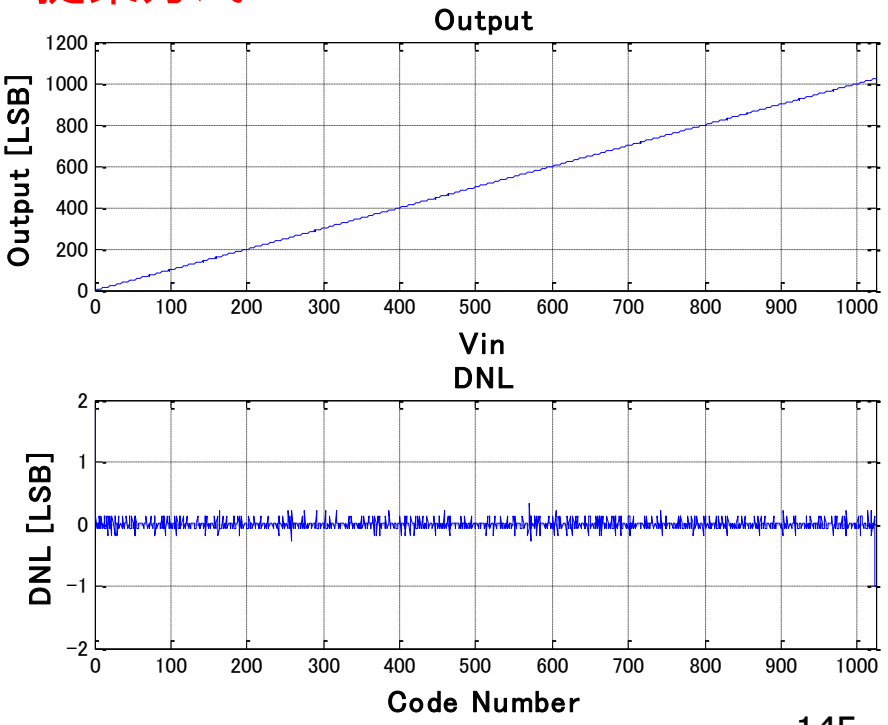
Comp1（低電力） オフセット：＋4.0 LSB、 ノイズ：1.0 LSB
Comp2（高電力） オフセット：－2.0 LSB、 ノイズ：0.2 LSB

コンパレータのアナログ・キャリブレーションなしの場合

IMEC方式



提案方式



消費電力とコンパレータミスマッチ許容のトレードオフ

非2進SAR ADC の特長

2進SAR ADC　　（IMEC: 西洋哲学）

構成，動作　無駄なし　→　効率的

冗長性（無駄）を入れる．

さらに効率が良くなる．

・スピード（DAC不完全整定）

・低消費電力化

冗長性により誤動作を許容

→各構成要素，動作への要求緩和

（「無用の用」老子: 東洋哲学）

147

内　容

- セミナーの目的・目標
- デジタル回路の基礎
- スイッチレベル　デジタルCMOS回路
- デジタルCMOS回路の性能（消費電力、スピード）
- 同期回路設計とカウンタ回路
- 加算器、ビットシフト、乗算器
- 事例1：　SAR ADC+分散型積和演算回路
- 事例2：　自己校正機能をもったTDC回路
- 事例3：　冗長アルゴリズムを用いたSAR ADC
- 最後に

デジタル技術の発展は産業・社会を変えた

- アナログ: 連続信号 「坂道」
  デジタル: 0, 1 「階段」
- デジタルは　産業的に
  技術のコピーを容易化
  ⇒ キャッチアップ早い
  インターフェースを容易化
  ⇒ エレクトロニクス産業の
  水平分業化 （産業構造が変わる）
- デジタルにより　社会的に
  人は数値で管理されるようになった

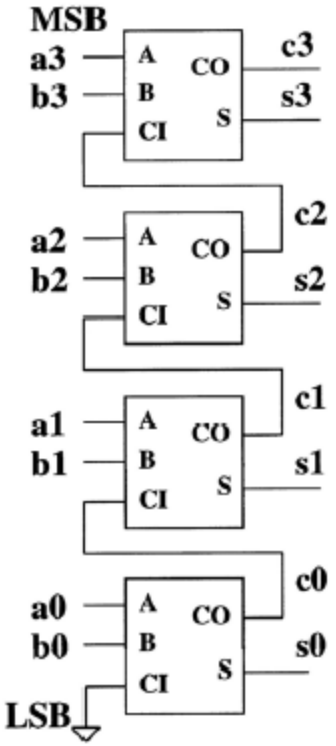
付録

Carry Look Ahead Adderとは

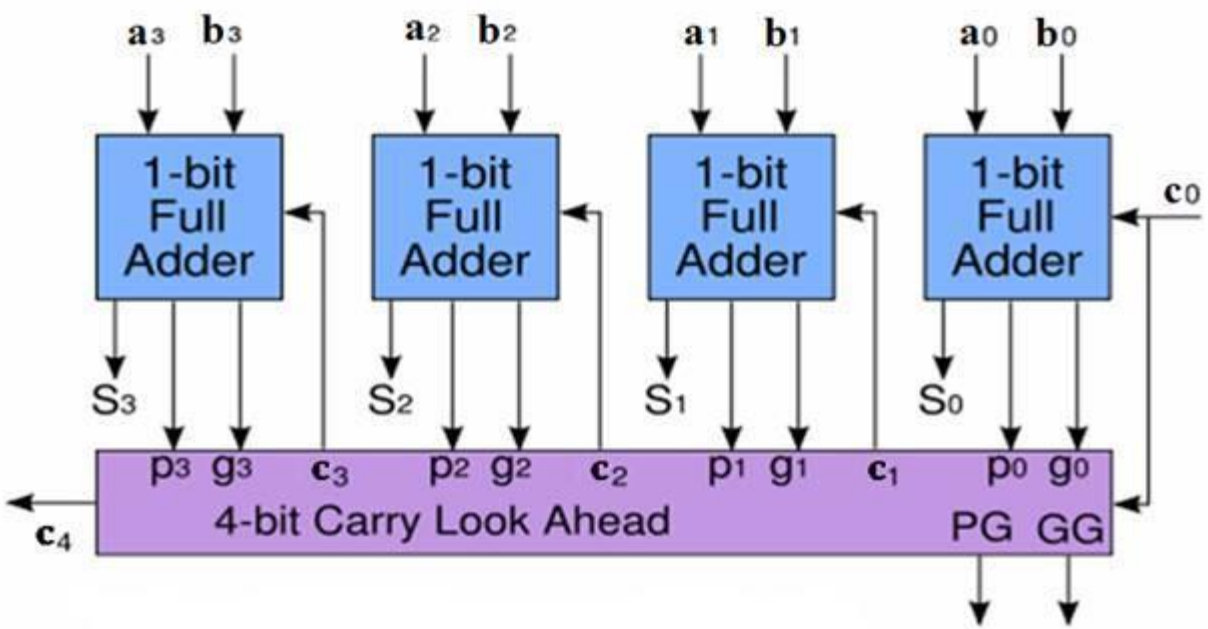
リップルキャリー加算器
下位ビットの桁上げが
次の上位ビットの入力に接続

桁数分のキャリーの伝播が生じるので
全体の加算演算の時間がかかる

桁上げ信号の部分
別に早く計算し高速化



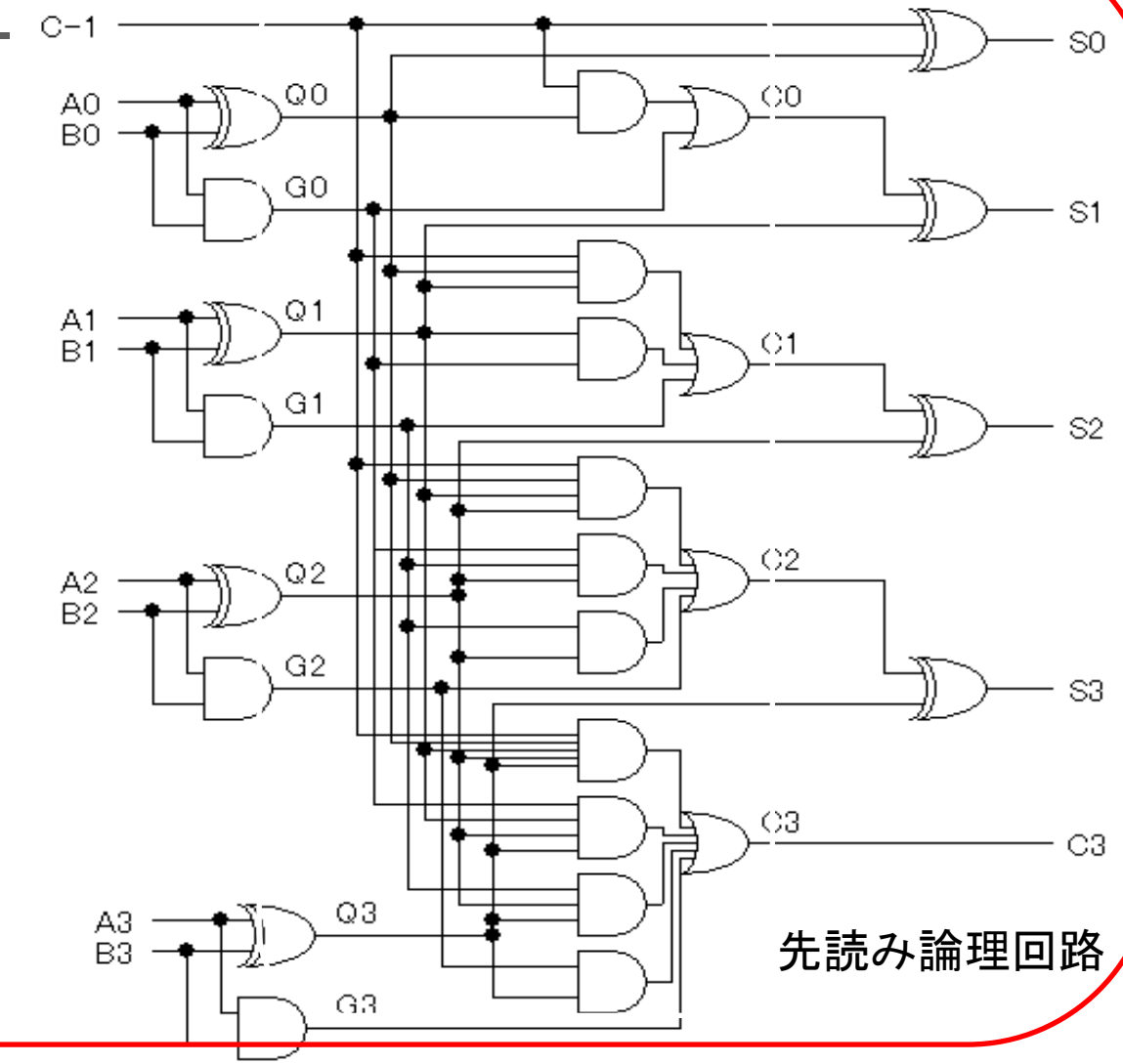
Carry Look Ahead Adderとは



桁上げ(Carry)を別に先に計算する

Carry Look Ahead Adder 回路

4bit Carry Look Ahead Adder



先読み論理回路

Carry Look Ahead 論理式

キャリー生成項 と キャリー伝播項

- Cn:nビット目のキャリー

$$C_n = A_n B_n + (A_n + B_n) \cdot C_{n-1}$$

$$= G_n + P_n \cdot C_{n-1}$$

- Gn:キャリー発生関数

$$G_n = A_n \cdot B_n$$

- Pn:キャリー伝播関数

$$P_n = A_n + B_n$$

Carry Look Ahead 論理式 (続き)

キャリーの計算

C0 = G0 + P0・C-1

これを、C1に代入

C1 = G1 + P1・C0

= G1 + P1・(G0 + P0・C-1)

= A1・B1 + (A1+B1){A0・B0 + (A0+B0)・C-1}

C0が決まればすぐにC1が決まる

Carry Look Ahead 論理式 (続き)

C2 = G2 + P2･C1
= G2 + P2･(G1 + P1･C0)
= G2 + P2･(G1 + P1･(G0 + P0･C-1))
= G2 + P2･G1 + P2･P1･G0 + P2･P1･P0･C-1

C2もC0が決まればすぐに決まる

C3 = G3 + P3･C2
= G3 + P3･(G2 + P2･(G1 + P1･(G0 + P0･C-1)))
= G3 + P3･G2 + P3･P2･G1 + P3･P2･P1･G0 + P3･P2･P1･P0･C-1

2020年5月12日, 19日（月）

集積回路システム演習問題
デジタルCMOS回路

群馬大学大学院

小林春夫

演習問題1： 論理演算

問題：グループ1の論理式の、グループ2の論理式と等しいものを示せ。

ヒント： 真理値表を書くと容易にわかる

グループ1： $A+A\cdot B$, $(A+B)\cdot B$

$A\cdot(\bar{A}+B)$, $A+\bar{A}\cdot B$

グループ2： A, B, $A+B$, $A\cdot B$

演習問題2: ドモルガンの法則

$$\overline{A \cdot B} = \bar{A} + \bar{B}, \quad \overline{A + B} = \bar{A} \cdot \bar{B}$$

$$\overline{A_1 + A_2 + \ldots + A_N} = \bar{A}_1 \cdot \bar{A}_2 \cdot \ldots \cdot \bar{A}_N$$

$$\bar{A}_1 + \bar{A}_2 + \ldots + \bar{A}_N = \overline{A_1 \cdot A_2 \cdot \ldots \cdot A_N}$$

問題: 上記の4つの式を証明せよ。

ヒント: N=2の場合は「真理値表」で示せ
一般のNの場合は数学的帰納法で示せ

演習問題3：　2進、8進、16進

問題1　　16進数で　AF5　を2進数で表せ。

問題2　　16進数で　5E8B を8進数で表せ。

問題3　　16進数で　27C　を10進数で表せ。

演習問題4: バイナリコードとグレイコード (1)

問題: G3, G2, G1, G0 を B3, B2, B1, B0 の論理式で表せ

例: G2 = B3⊕B2

decimal	Binary code				Gray code			
Data	B3	B2	B1	B0	G3	G2	G1	G0
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	0	1	0
4	0	1	0	0	0	1	1	0
5	0	1	0	1	0	1	1	1
6	0	1	1	0	0	1	0	1
7	0	1	1	1	0	1	0	0
8	1	0	0	0	1	1	0	0
9	1	0	0	1	1	1	0	1
10	1	0	1	0	1	1	1	1
11	1	0	1	1	1	1	1	0
12	1	1	0	0	1	0	1	0
13	1	1	0	1	1	0	1	1
14	1	1	1	0	1	0	0	1
15	1	1	1	1	1	0	0	0

演習問題5: バイナリコードとグレイコード (2)

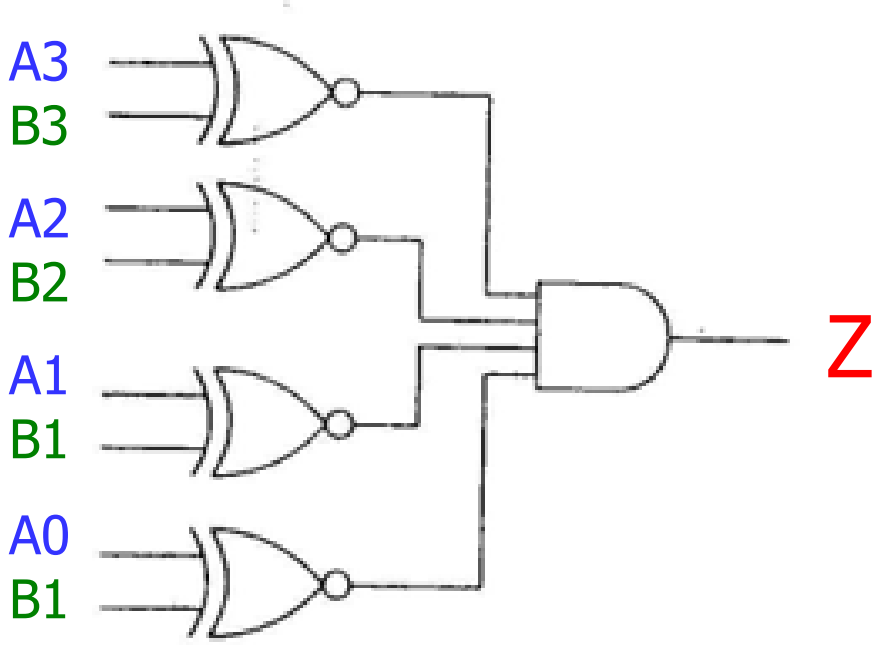
問題: B3, B2, B1, B0 を G3, G2, G1, G0 の論理式で表せ

例: B2 = G3⊕G2

decimal	Binary code				Gray code			
Data	B3	B2	B1	B0	G3	G2	G1	G0
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	0	1	0
4	0	1	0	0	0	1	1	0
5	0	1	0	1	0	1	1	1
6	0	1	1	0	0	1	0	1
7	0	1	1	1	0	1	0	0
8	1	0	0	0	1	1	0	0
9	1	0	0	1	1	1	0	1
10	1	0	1	0	1	1	1	1
11	1	0	1	1	1	1	1	0
12	1	1	0	0	1	0	1	0
13	1	1	0	1	1	0	1	1
14	1	1	1	0	1	0	0	1
15	1	1	1	1	1	0	0	0

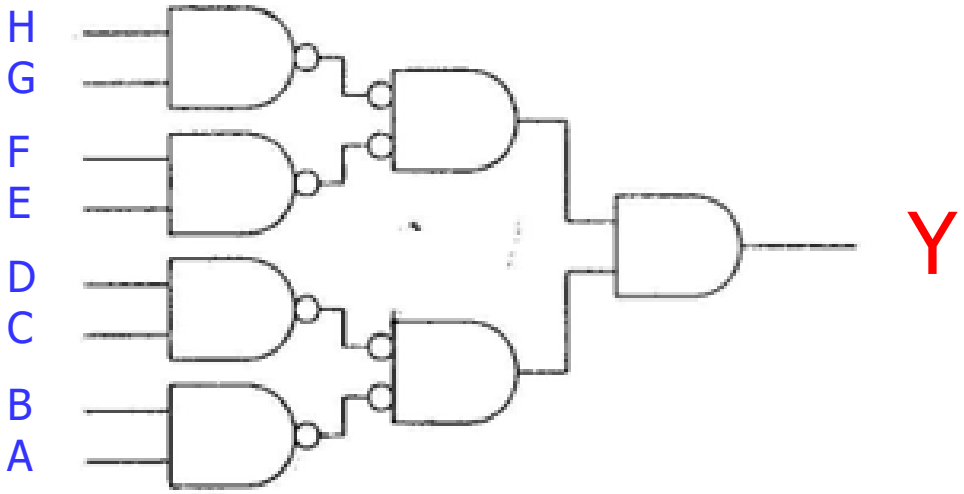
演習問題6: 論理回路

問題: Z=1 になるときの
入力 A3, A2, A1, A0, B3, B2, B1, B0 の条件を示せ



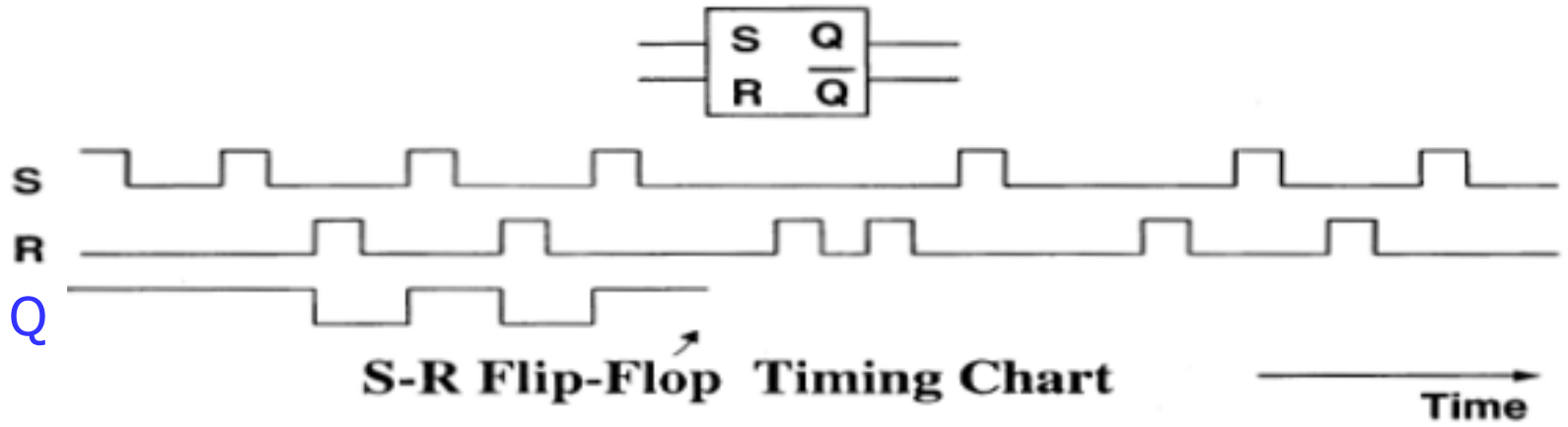
演習問題7:　論理回路

問題:　Y=1 になるときの
　入力　A, B, C, D, E, F, G, H の条件を示せ



演習問題8： SR フリップフロップ

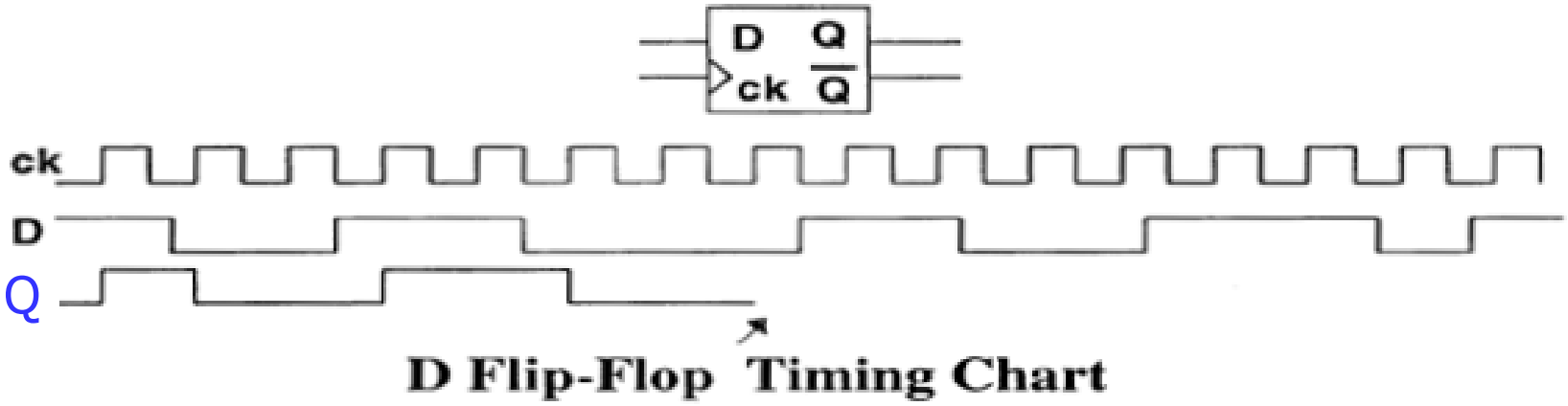
問題:　次の SRフリップフロップ(Set-Reset Flip-Flop)
のタイミングチャートを描け



Q の波形を描く

演習問題9： D フリップフロップ

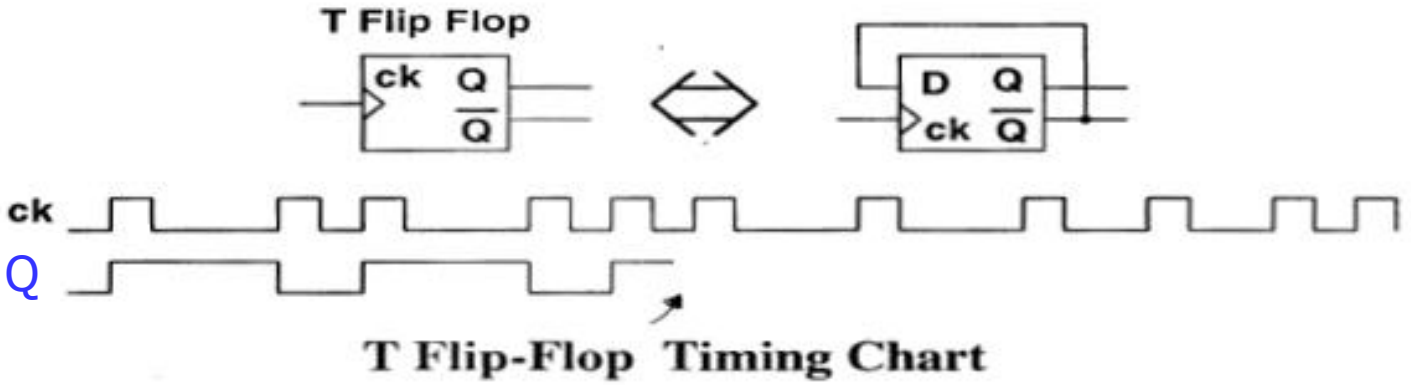
問題:　次の Dフリップフロップ(D Flip-Flop)
のタイミングチャートを描け



Q の波形を描く

演習問題10： T フリップフロップ

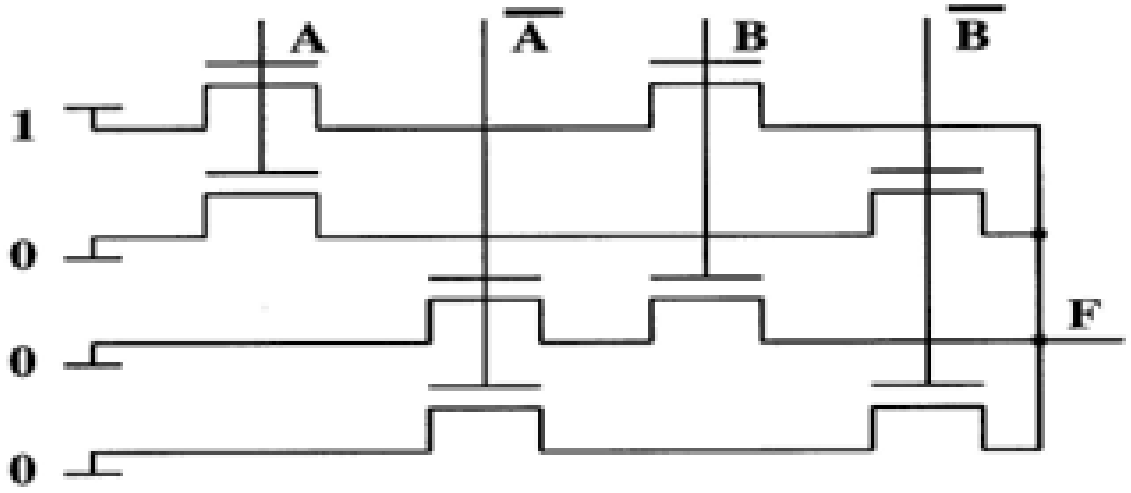
問題:　次の T フリップフロップ(Toggle Flip-Flop)
のタイミングチャートを描け



Q の波形を描く

演習問題：NMOS論理回路 （例）

問題:　 Zを A, Bを用いた論理式を書け

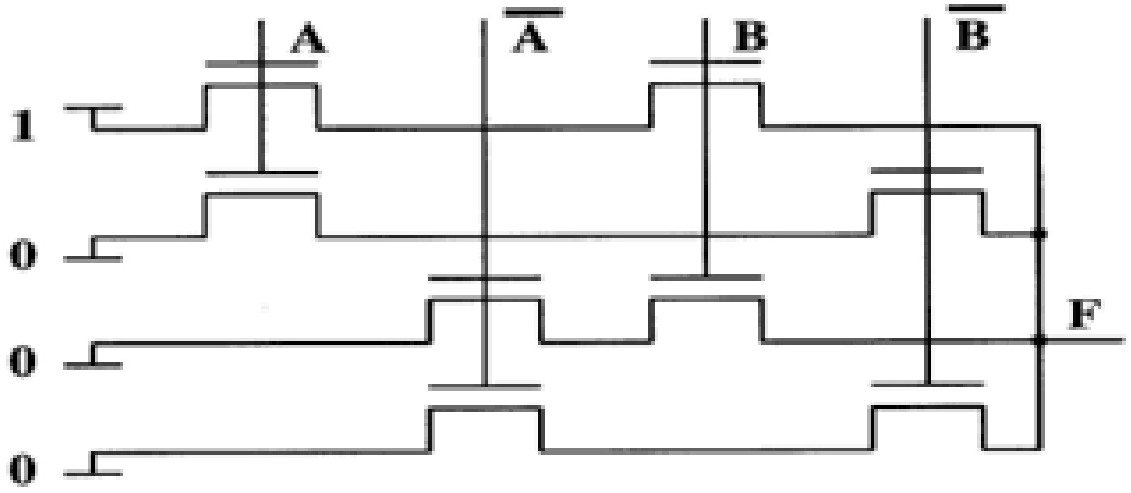


解答

$F = A\ B$

演習問題11： NMOS論理回路1

問題:　Zを A, Bを用いた論理式を書け

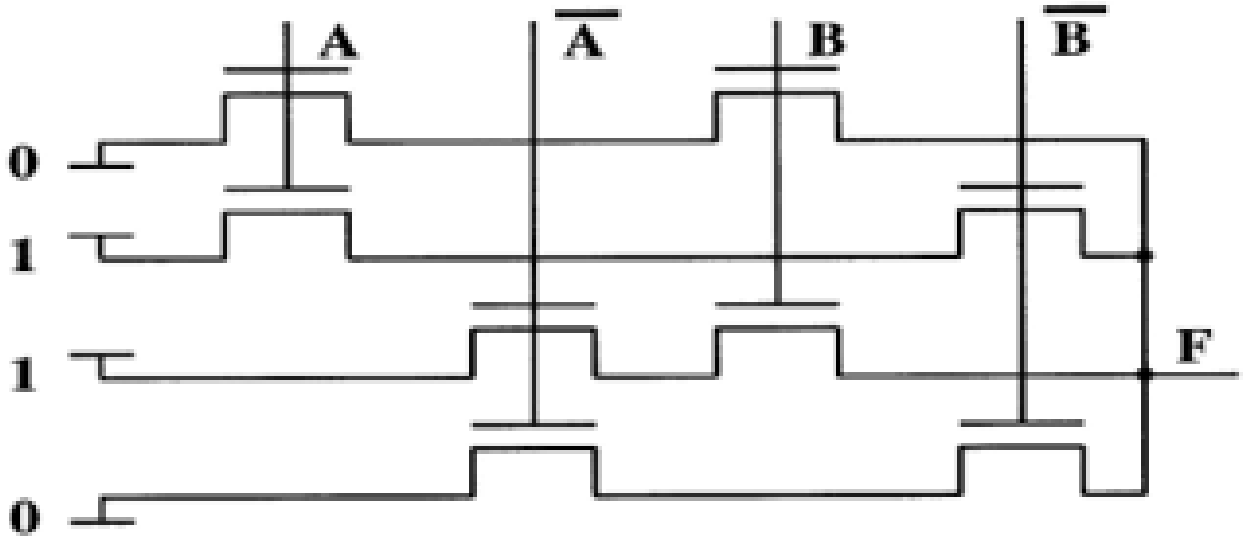


解答

$F = A\ B$

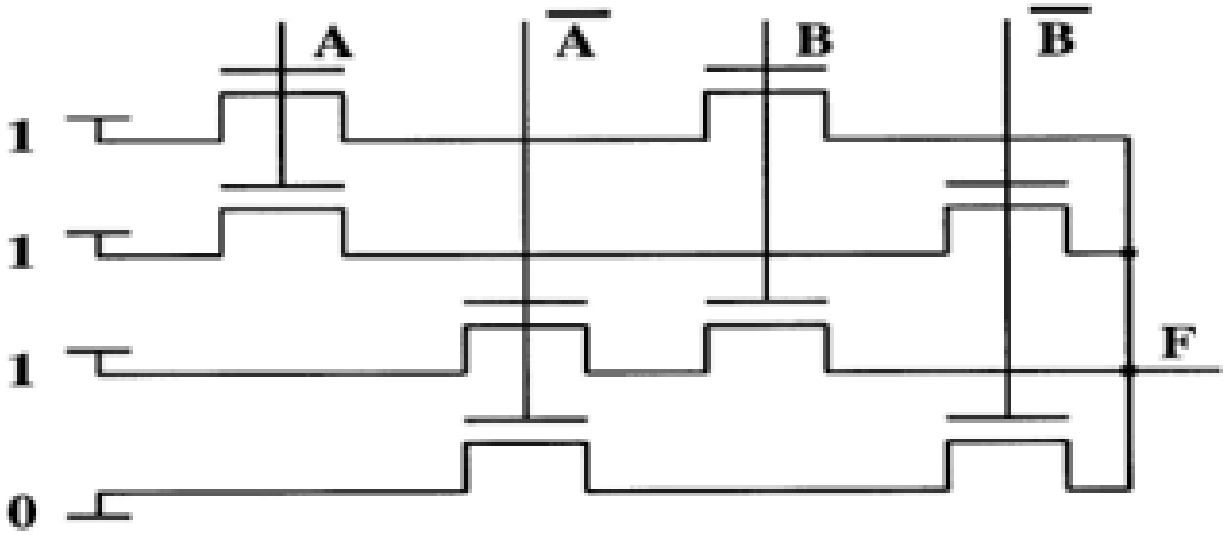
演習問題12： NMOS論理回路 2

問題:　Zを A, Bを用いた論理式を書け



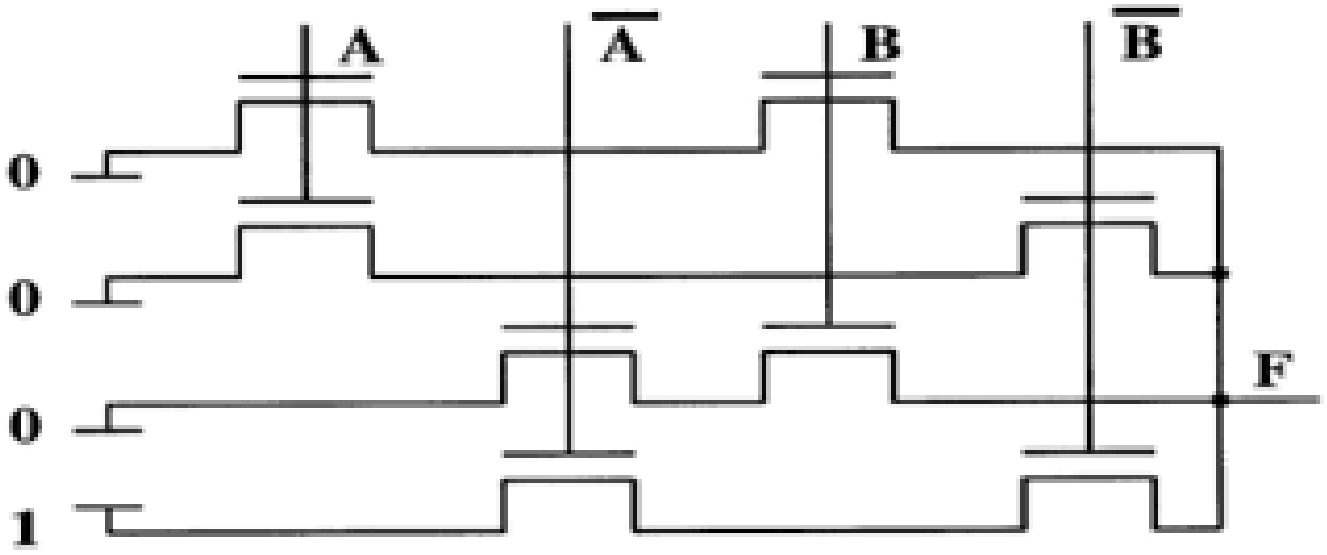
演習問題13： NMOS論理回路 3

問題:　Zを A, Bを用いた論理式を書け



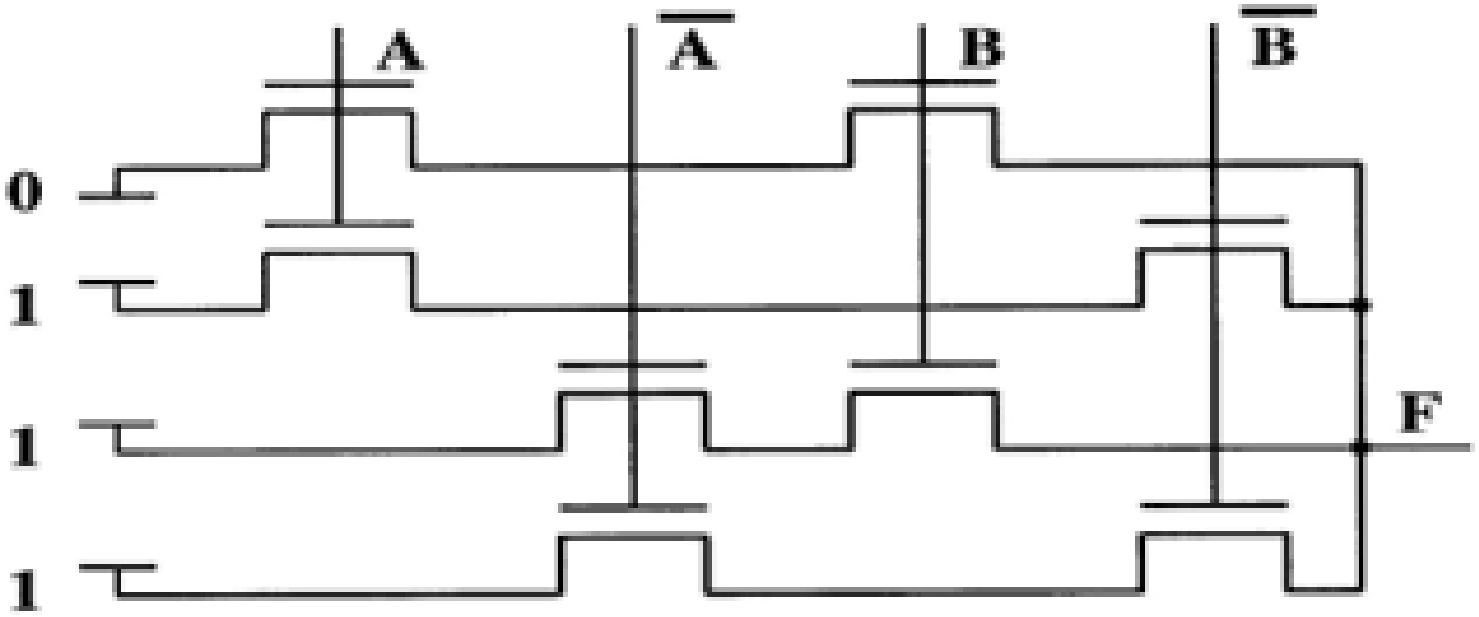
演習問題14： NMOS論理回路 4

問題:　Zを A, Bを用いた論理式を書け



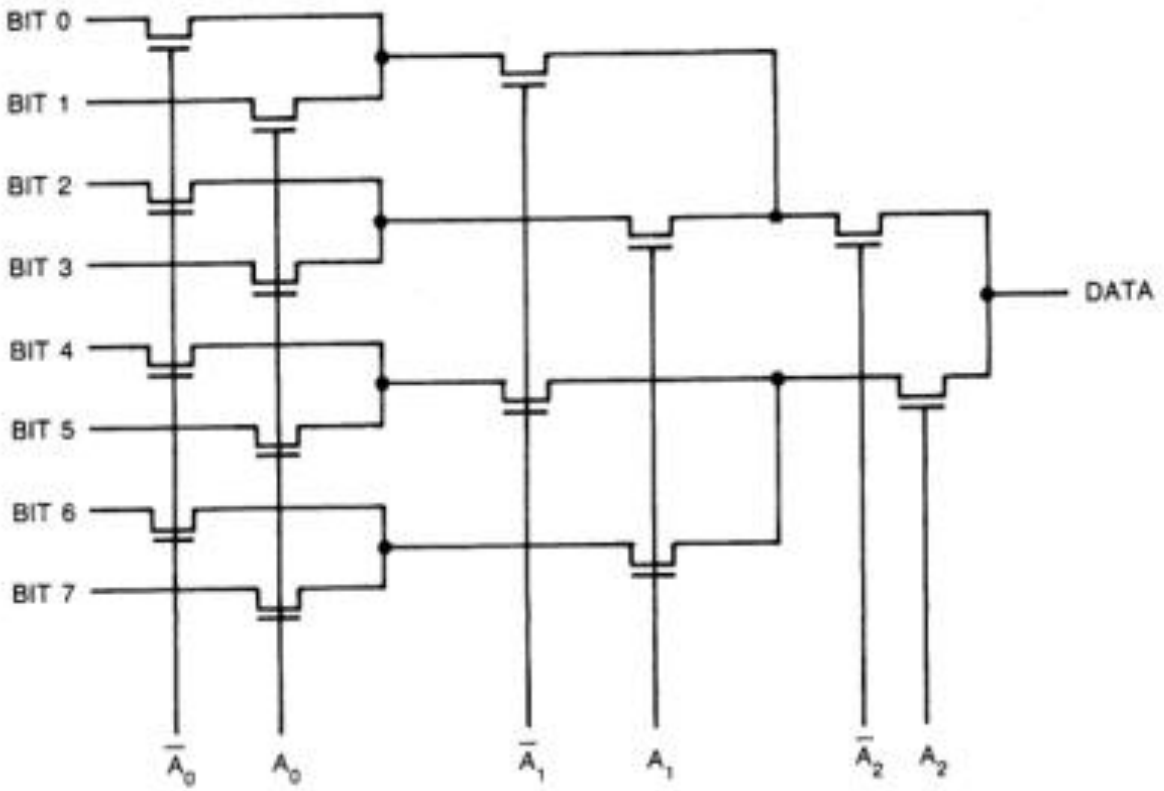
演習問題15： NMOS論理回路 5

問題:　Zを A, Bを用いた論理式を書け



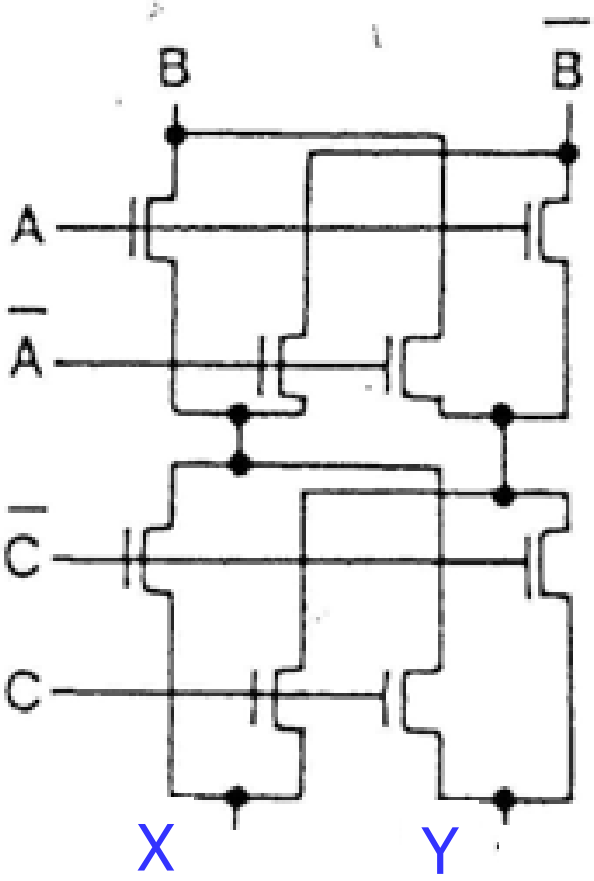
演習問題16： NMOS論理回路 6

問題:　DATAを A2, A1, A0, BIT7, BIT6,.., BIT0 の論理式で表せ



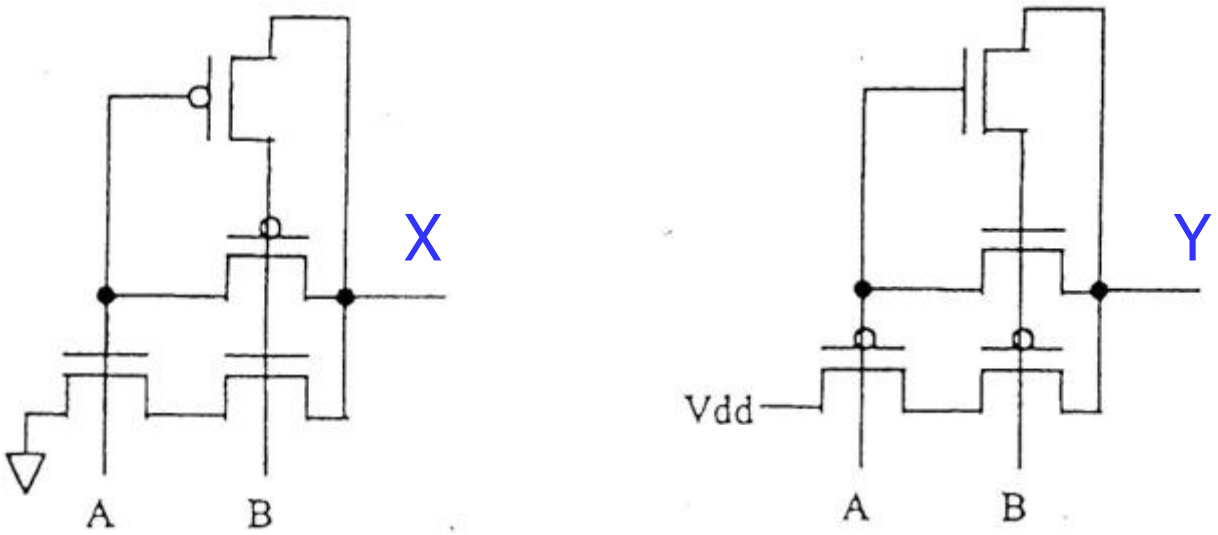
演習問題17： NMOS論理回路 7

問題:　X, Y を A, B, C を用いた論理式を書け



演習問題18:
NMOS,PMOS論理回路 8

問題:　X, Y を A, B を用いた論理式を書け



演習問題19:
3入力　EXOR回路

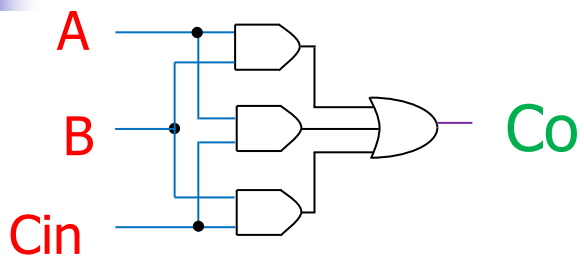
$S = (A \oplus B) \oplus Cin$

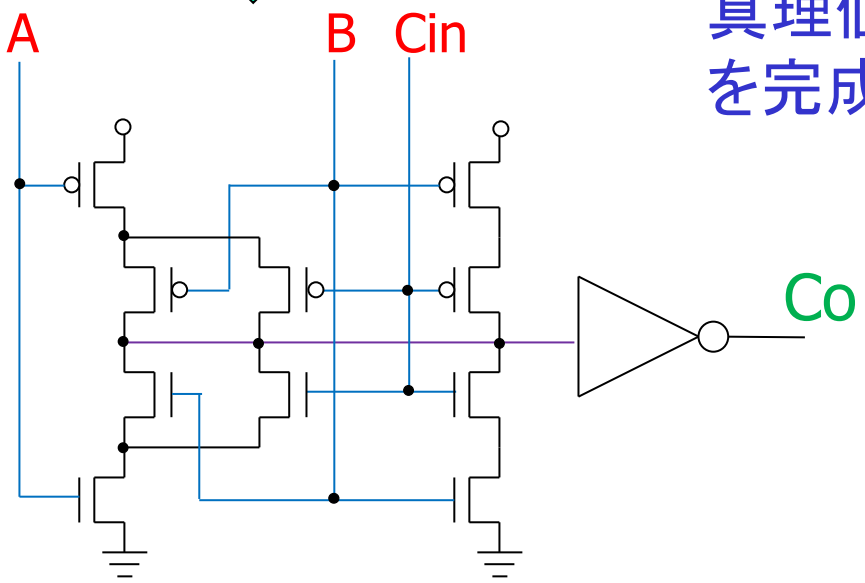
真理値表
を完成せよ

0, 1 を埋めよ

A	B	Cin	S
0	0	0	
0	0	1	
0	1	0	
1	0	0	
0	1	1	
1	0	1	
1	1	0	
1	1	1	

演習問題20：
多数決回路 (Majority Circuit)





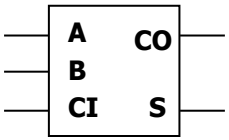
真理値表
を完成せよ

0, 1 を埋めよ

A	B	Cin	Co
0	0	0	
0	0	1	
0	1	0	
1	0	0	
0	1	1	
1	0	1	
1	1	0	
1	1	1	

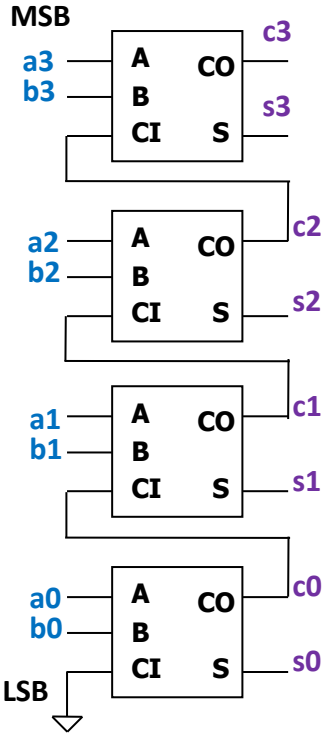
演習問題21：
全加算器と桁上げ伝播

Full Adder



A	B	CI	S	CO
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
1	0	0	1	0
0	1	1	0	1
1	1	0	0	1
1	0	1	0	1
1	1	1	1	1

Adder & Carry Propagation



(a3 a2 a1 a0) =(0 1 0 1)
(b3 b2 b1 b0)=(1 0 1 1)

のとき

(s3 s2 s1 s0)
(c3 c2 c1 c0)

を左の図から求めよ

演習問題22：
8bit 桁上げ伝播加算器
(Carry Ripple Adder)

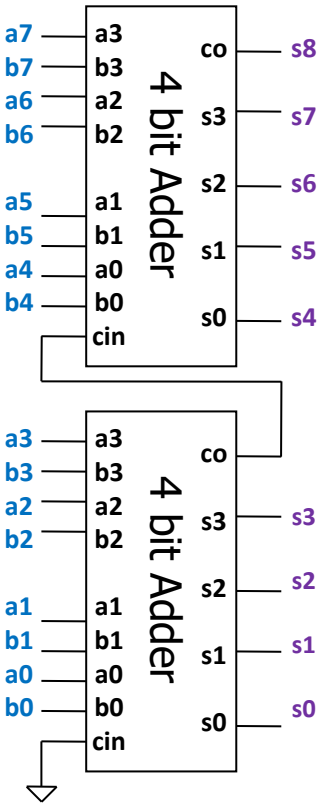
(a7 a6 a5 a4 a3 a2 a1 a0) =
(1 0 1 1 0 1 0 1)

(b7 b6 b5 b4 b3 b2 b1 b0)=
(0 0 1 1 1 0 1 1)

のとき

(s8 s7 s6 s5 s4 s3 s2 s1 s0)
の　各　0, 1 の値を得よ

この回路方式の
デメリットを述べよ。

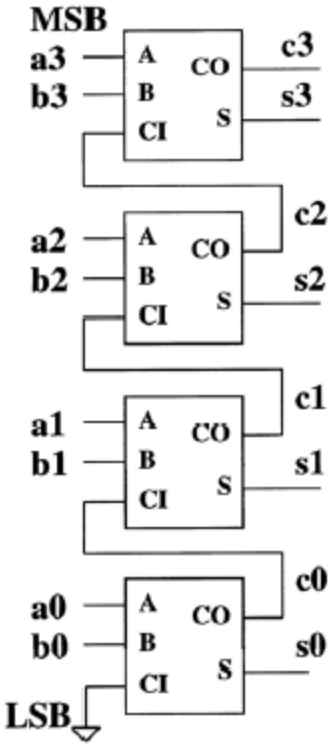


桁上げ伝搬加算器の問題点

リップルキャリー加算器
下位ビットの桁上げが
次の上位ビットの入力に接続

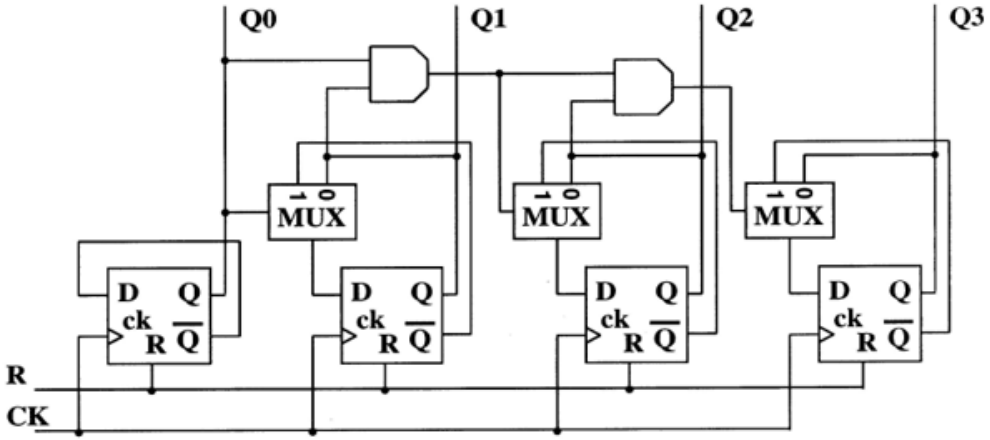
桁数分のキャリーの伝播が生じるので
全体の加算演算の時間がかかる

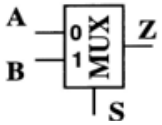
桁上げ信号の部分
別に早く計算し高速化



問題24　カウンタ (1)

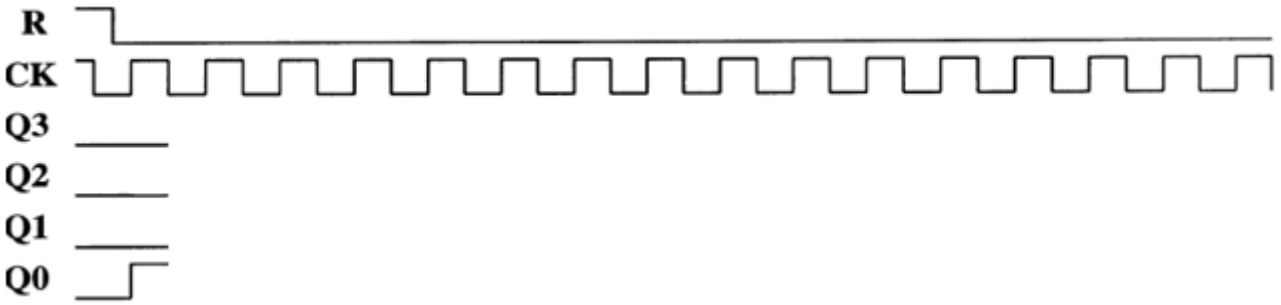
次の回路のタイミングチャートを完成させよ





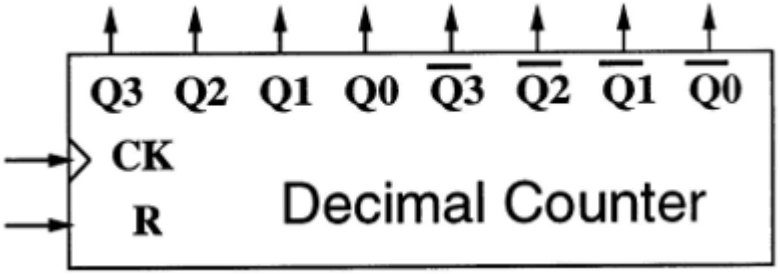
when S=0, Z=A

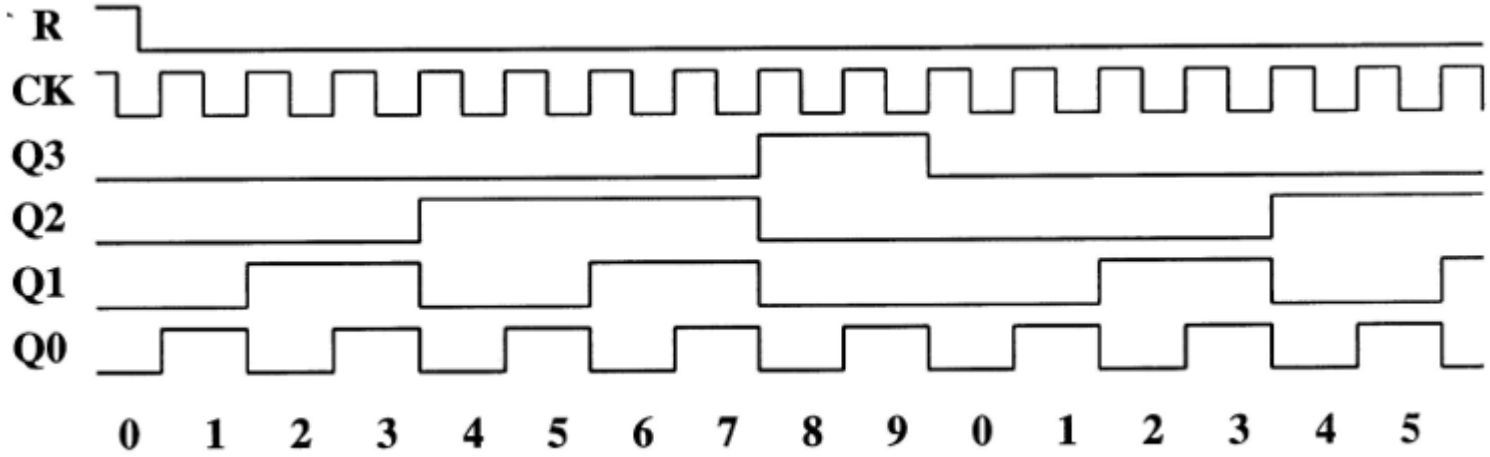
when S=1, Z=B



問題25　カウンタ (2)

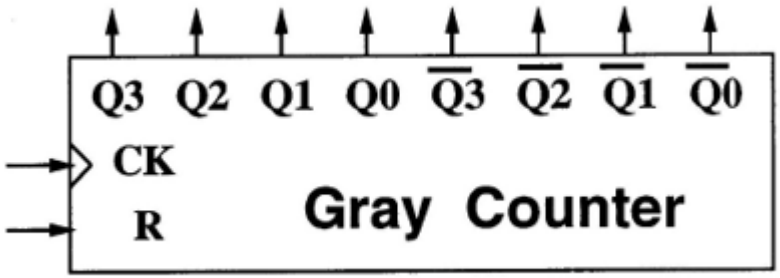
次の10進カウンタ(Decimal Counter) 回路を設計せよ

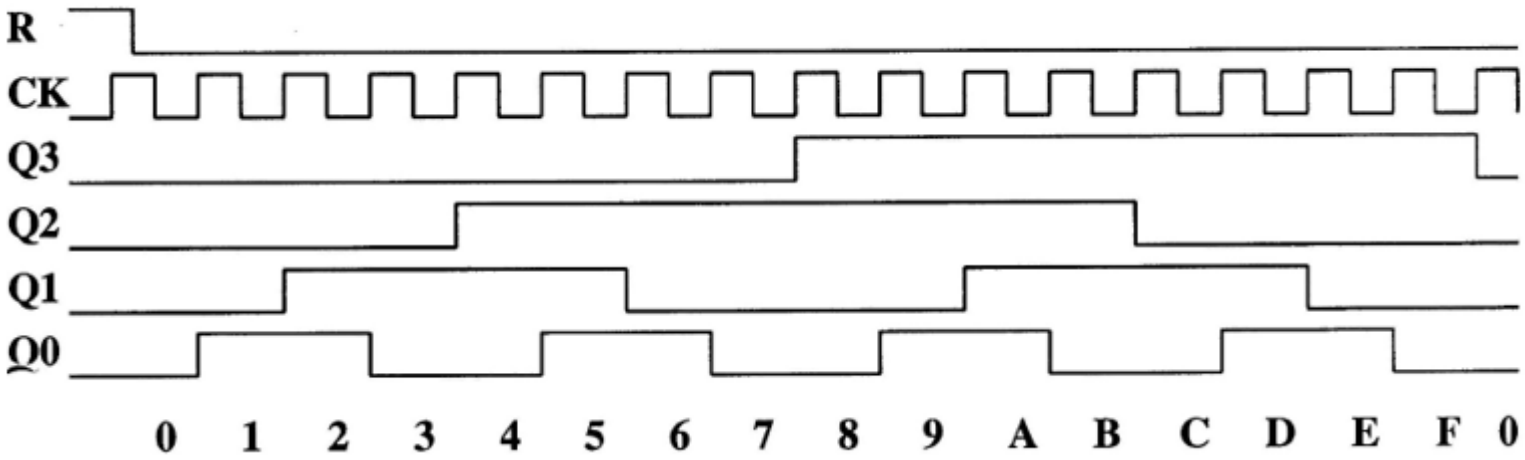




問題26　カウンタ (3)

次のグレイ カウンタ(Gray Counter) 回路を設計せよ

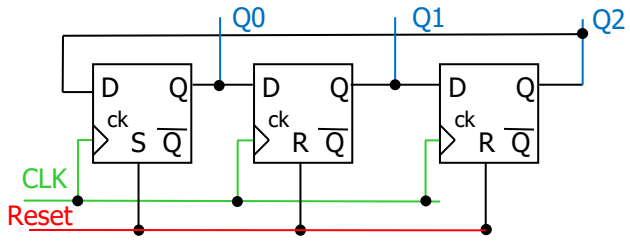


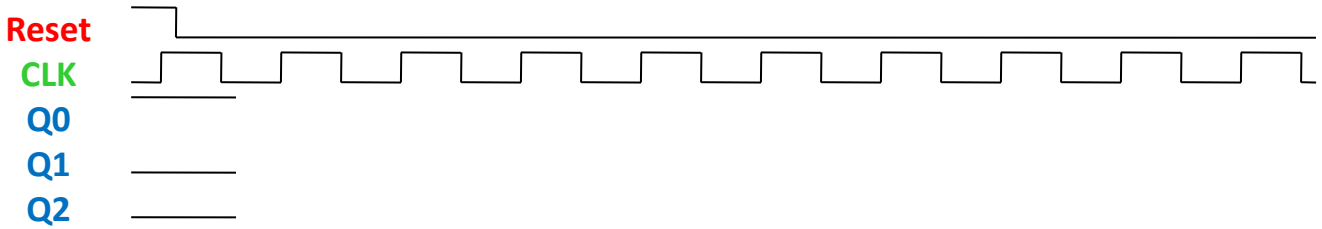


Gray code を出力するカウンタ

問題27　リング･カウンタ　（Ring Counter)

次の回路のタイミングチャートを完成させよ

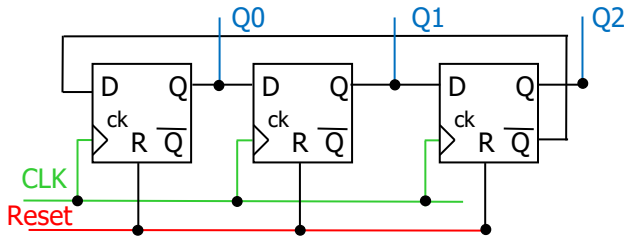


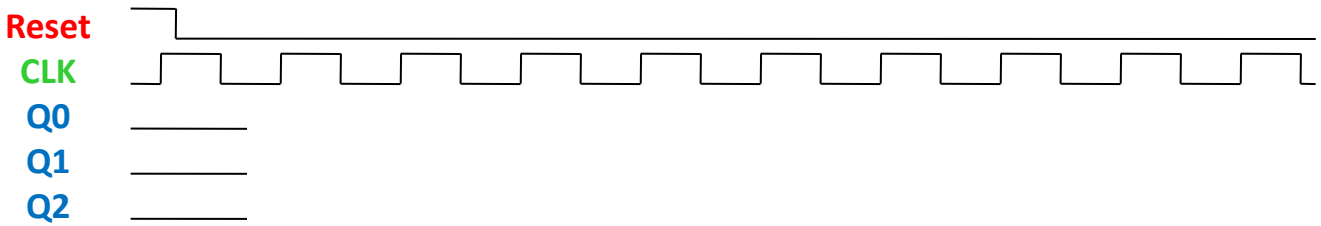


逐次比較近似ADCのタイミング発生回路等に使用される。

問題28 ジョンソン・カウンタ （Johnson Counter)

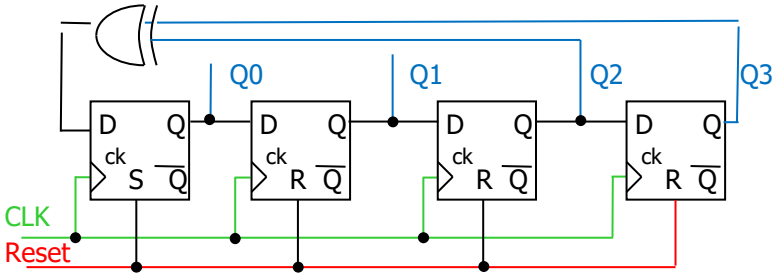
次の回路のタイミングチャートを完成させよ



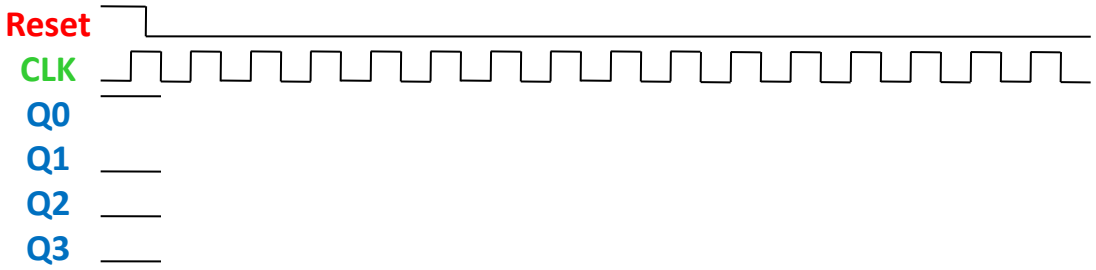


リングカウンタ回路と似ているが
出力信号は大きく異なる。
デジタル計算機のシーケンサ回路として使用された。

問題29　疑似ランダム信号発生回路 (Linear Feedback Shift Register)



←結線に注意！



- Q0=Q1=Q2=Q3=0 以外の15通りの信号を発生
- （再現性のある）疑似ランダム信号を発生
- フィードバックの取り方には決まりあり

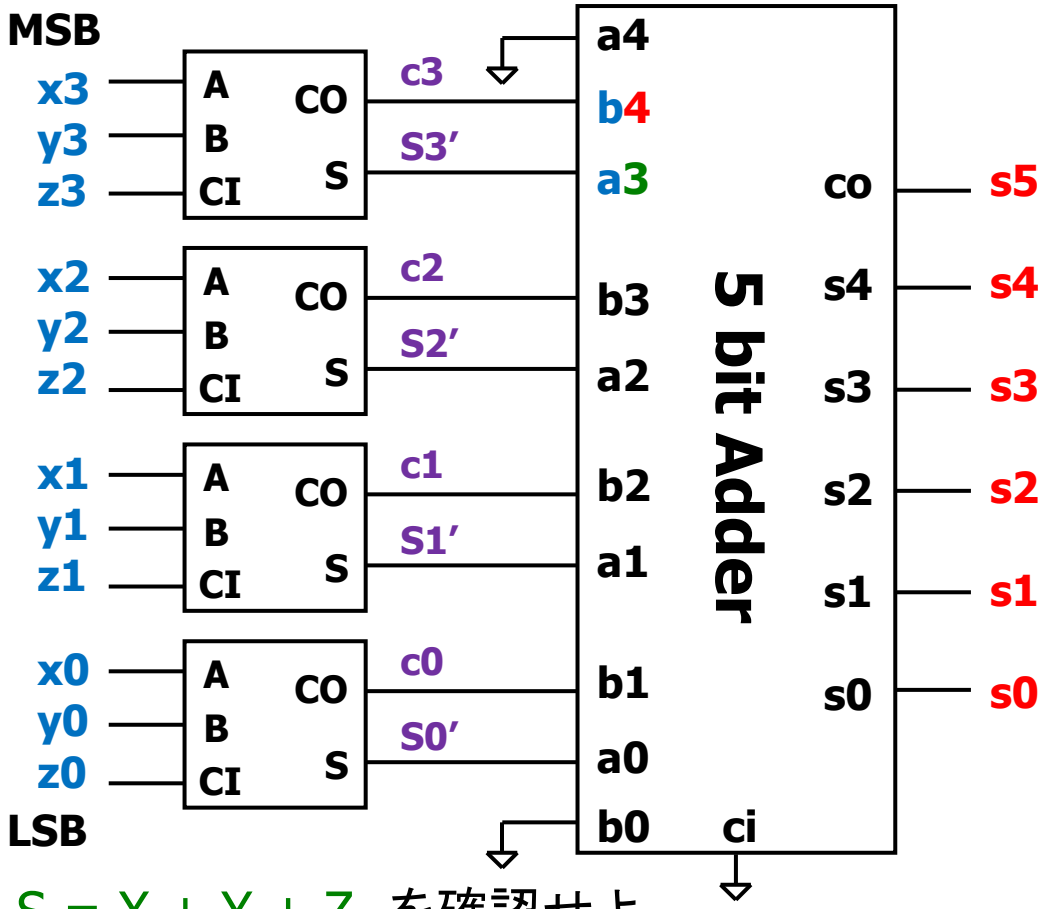
問題30　桁上げ計算節約加算器
Carry Save Adder

(x3 x2 x1 x0)=
(　1　0　 1　1)

(y3 y2 y1 y0)=
(　0　1　 1　1)

(z3 z2 z1 z0)=
(　1　1　 1　0)
のとき

(c3 c2 c1 c0)
(s3′ s2′ s1′ s0′)
(s5 s4 s3 s2 s1 s0)
の値を求めよ

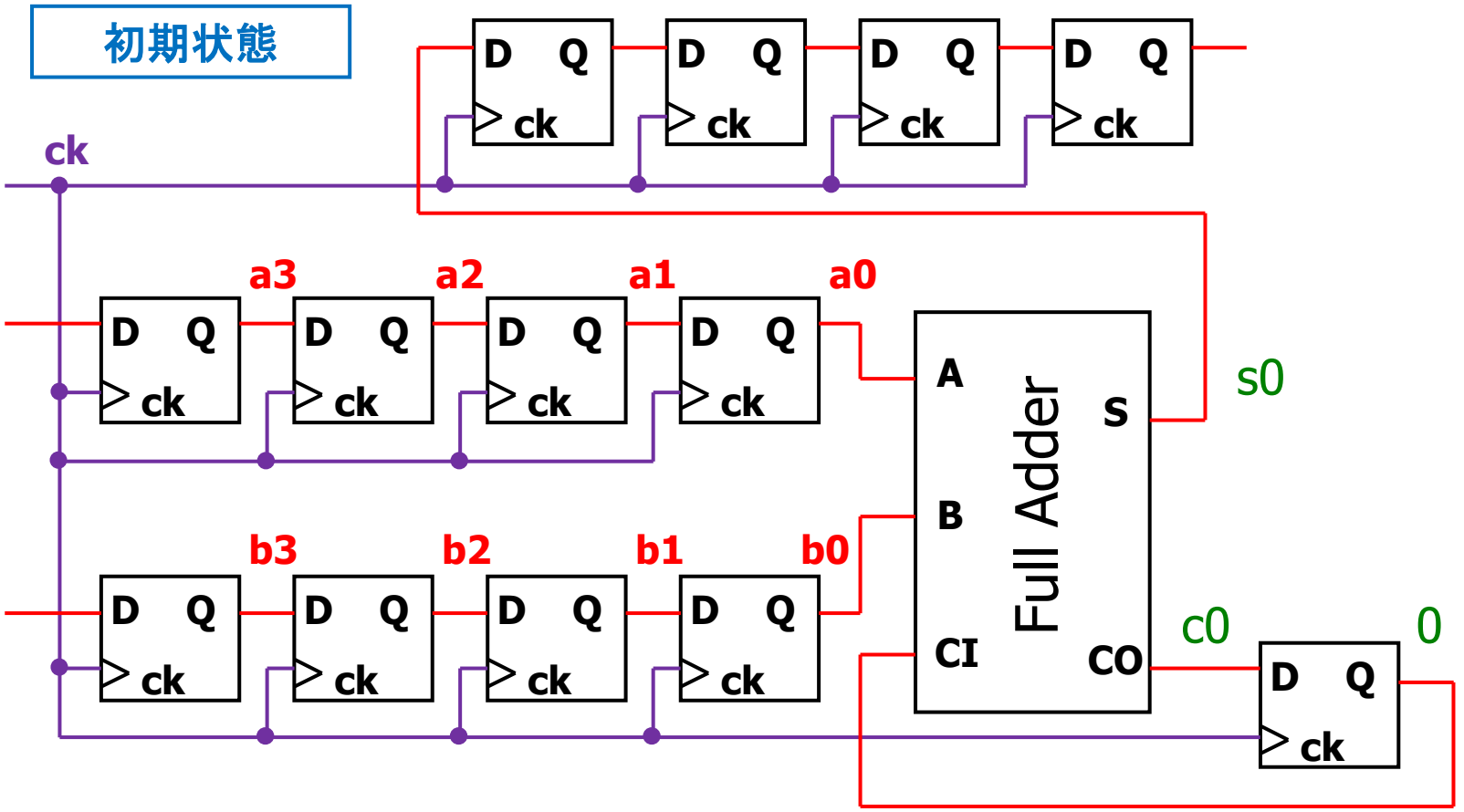


X, Y, Z, S の10進表現で S = X + Y + Z を確認せよ

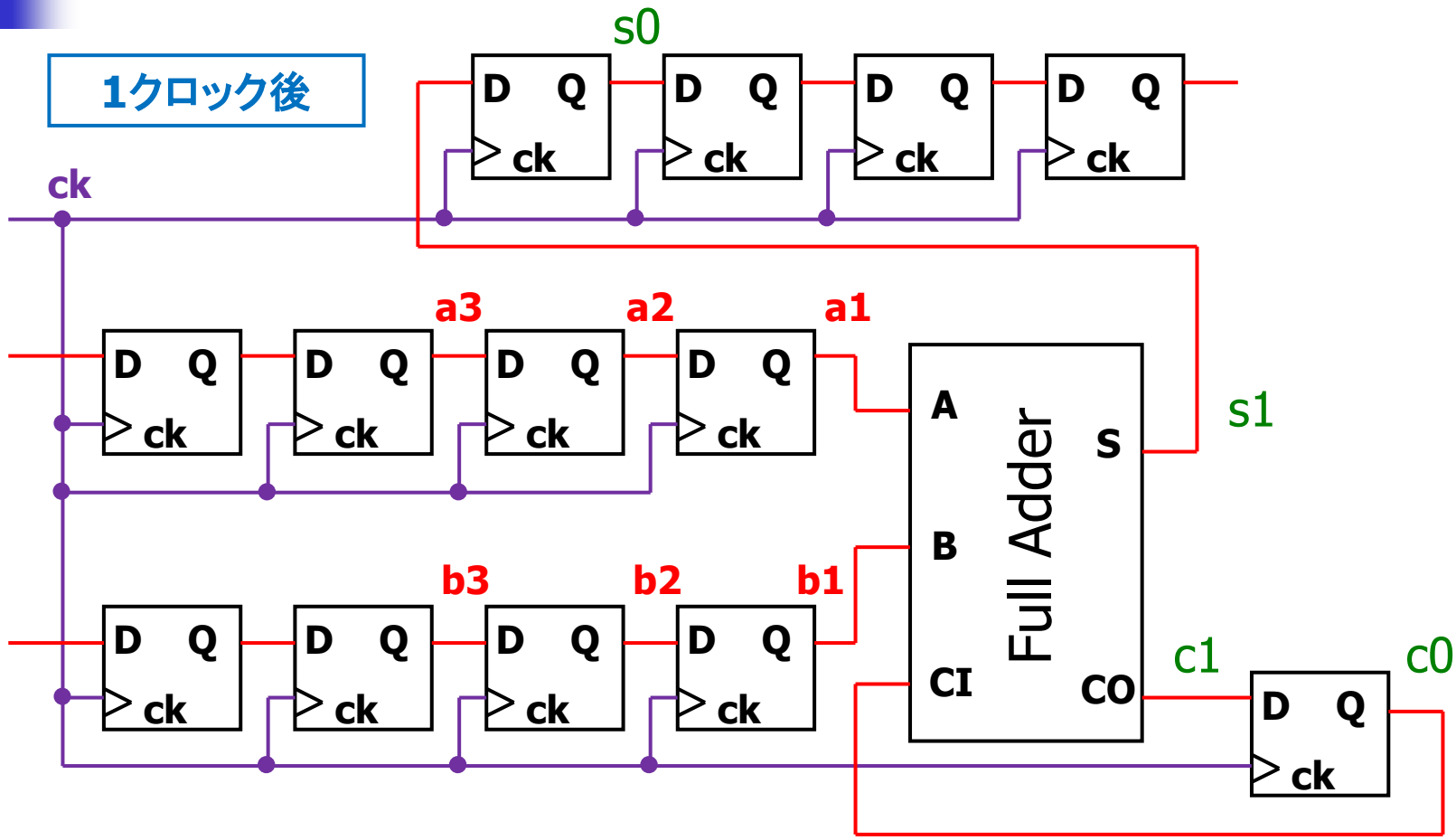
この回路のメリットは何か述べよ

問題31 ビットシリアル加算器

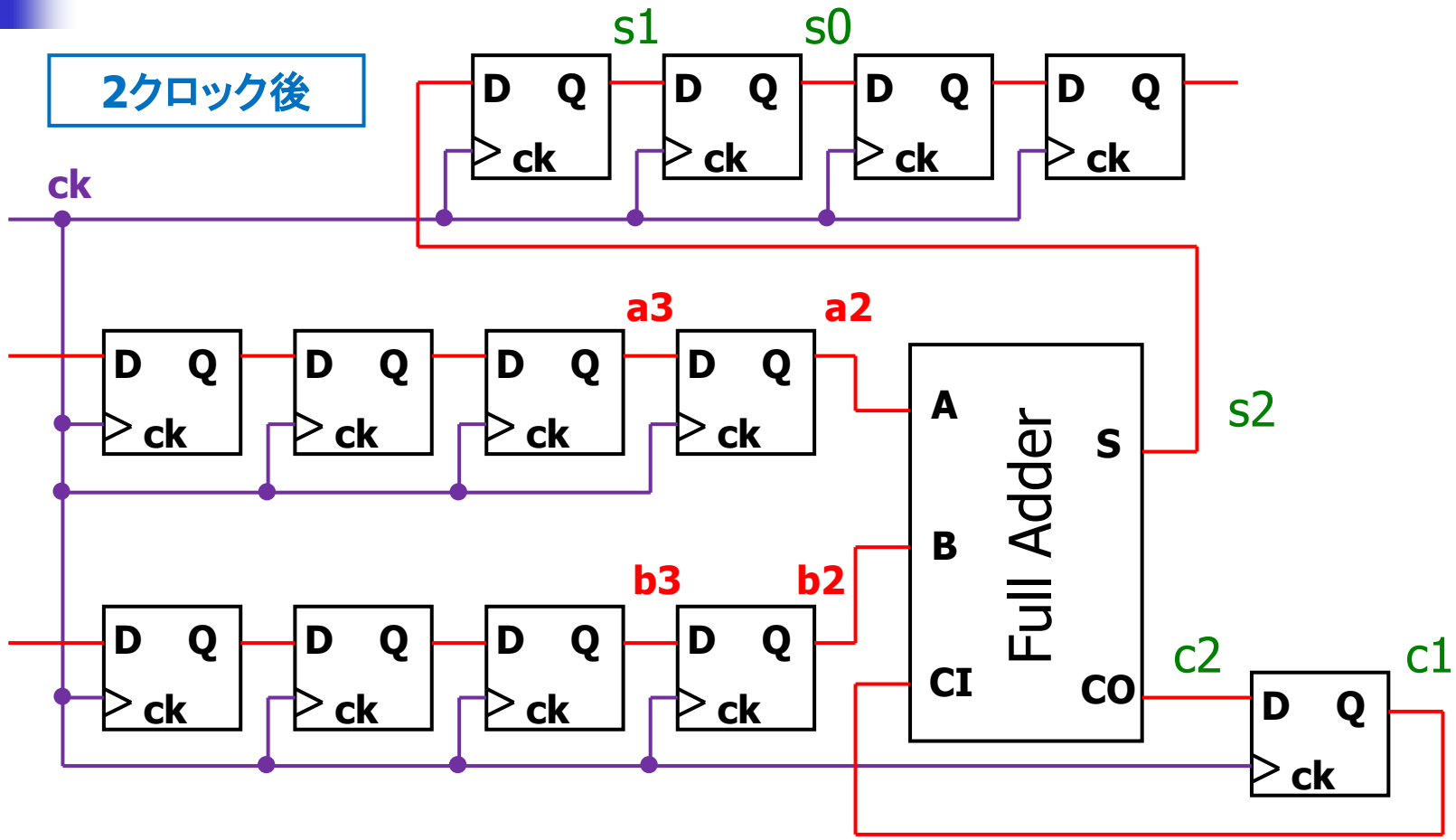
(a3 a2 a1 a0)=(1 0 1 0), (b3 b2 b1 b0)=(1 1 1 0) のとき
(s3 s2 s1 s0), (c3 c2 c1 c0) の値を求めよ



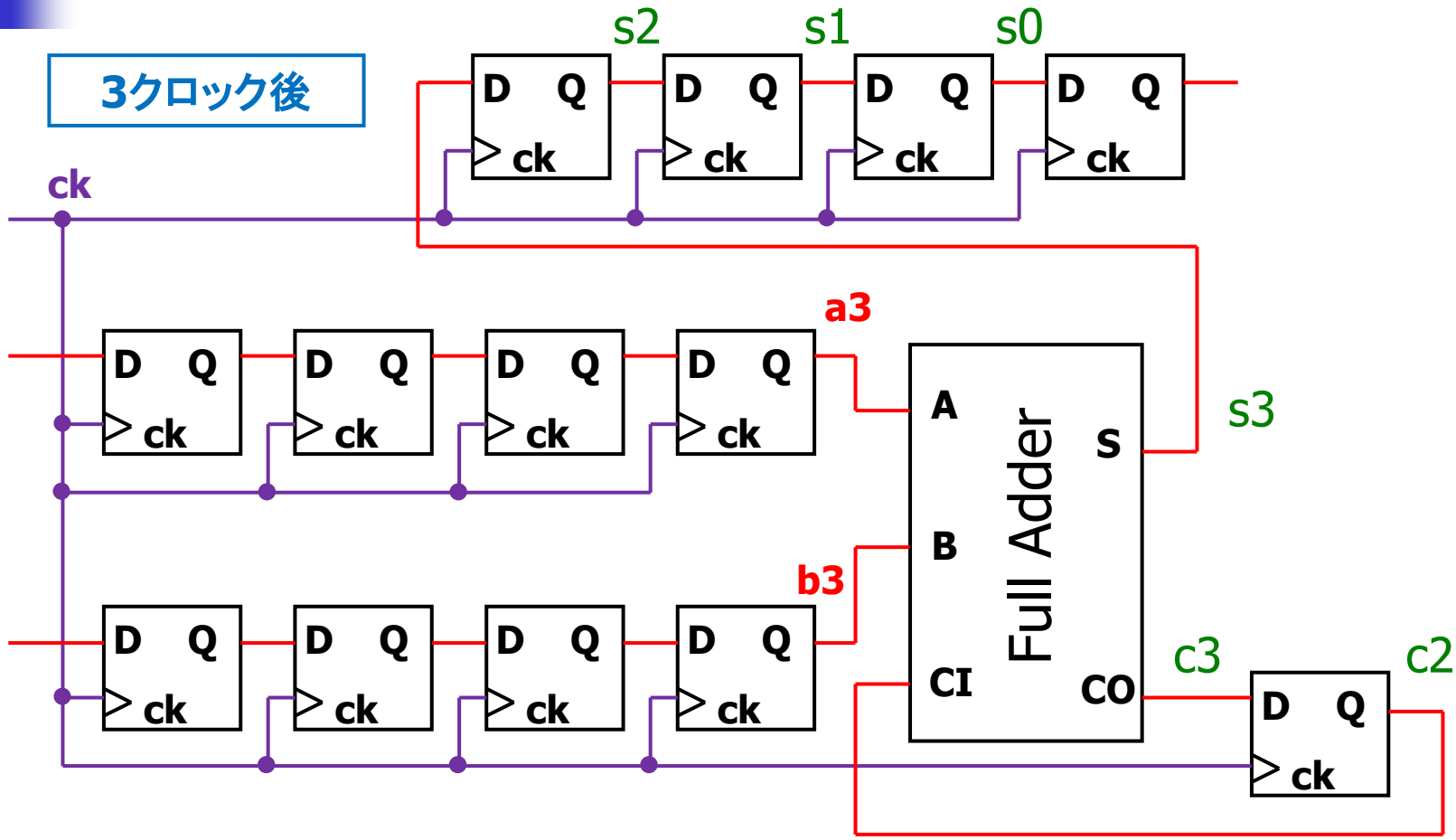
ビットシリアル加算器の動作



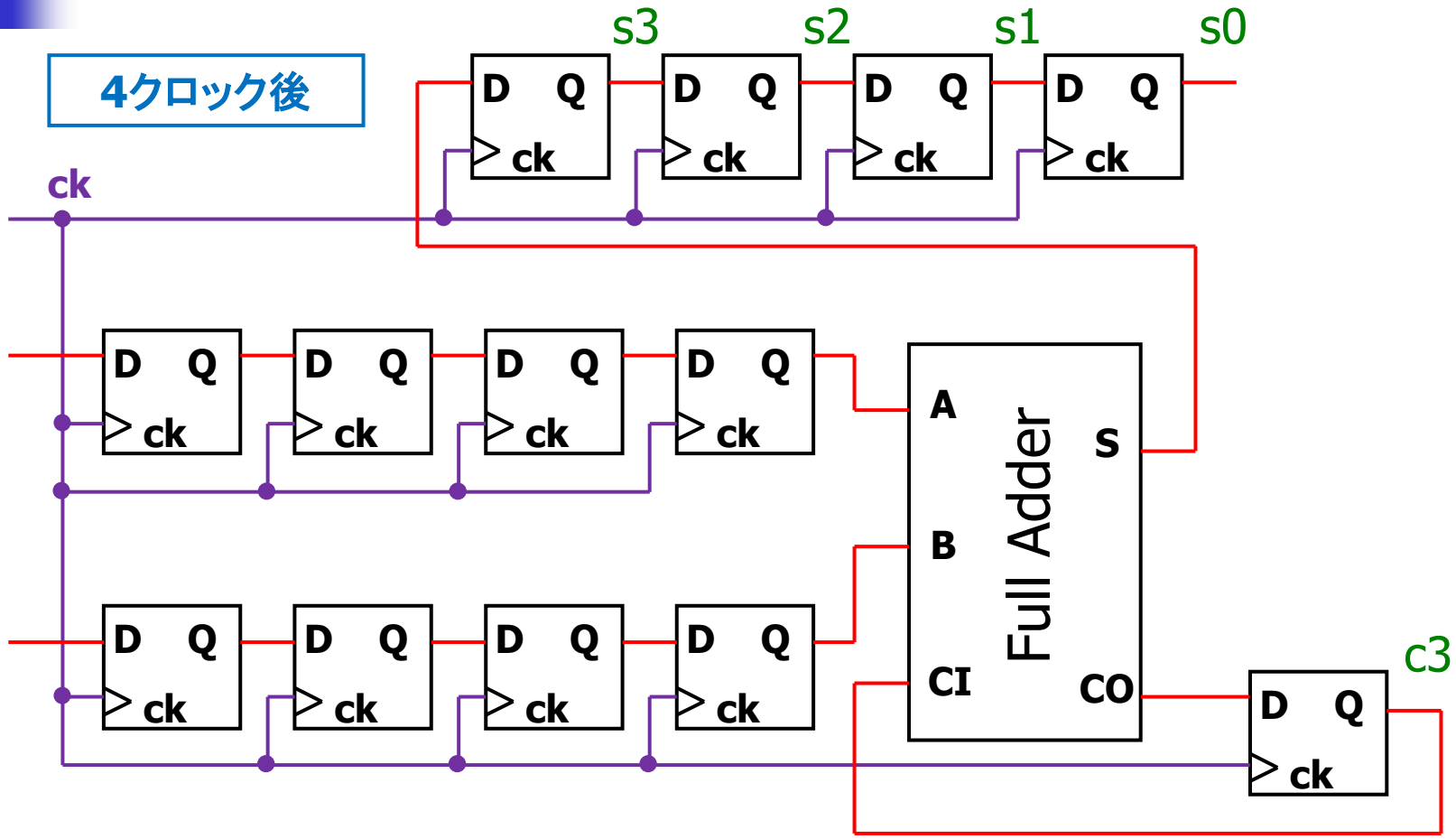
ビットシリアル加算器の動作

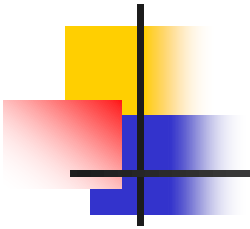


ビットシリアル加算器の動作



ビットシリアル加算器の動作



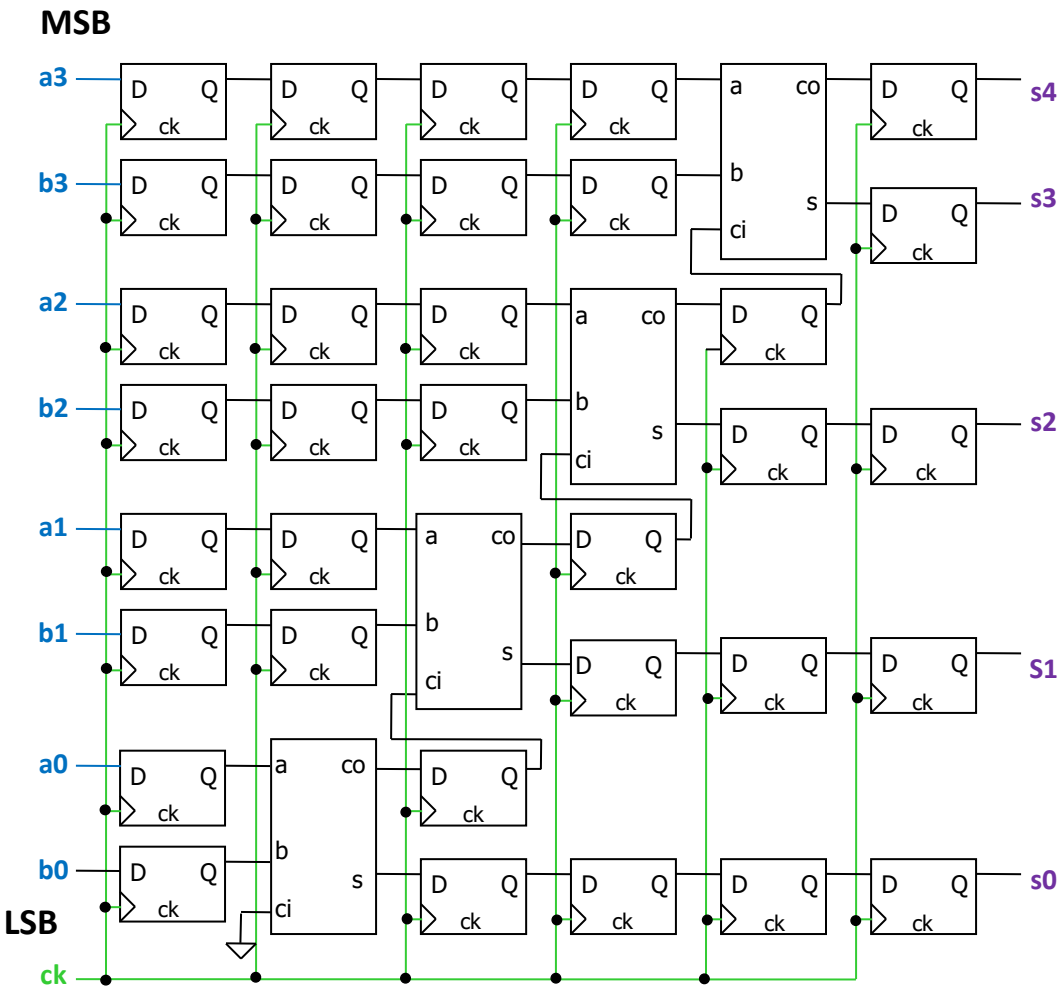


問題32 ビットシリアル加算器

ビットシリアル加算器の
メリット、デメリットを述べよ

問題33 パイプライン加算器

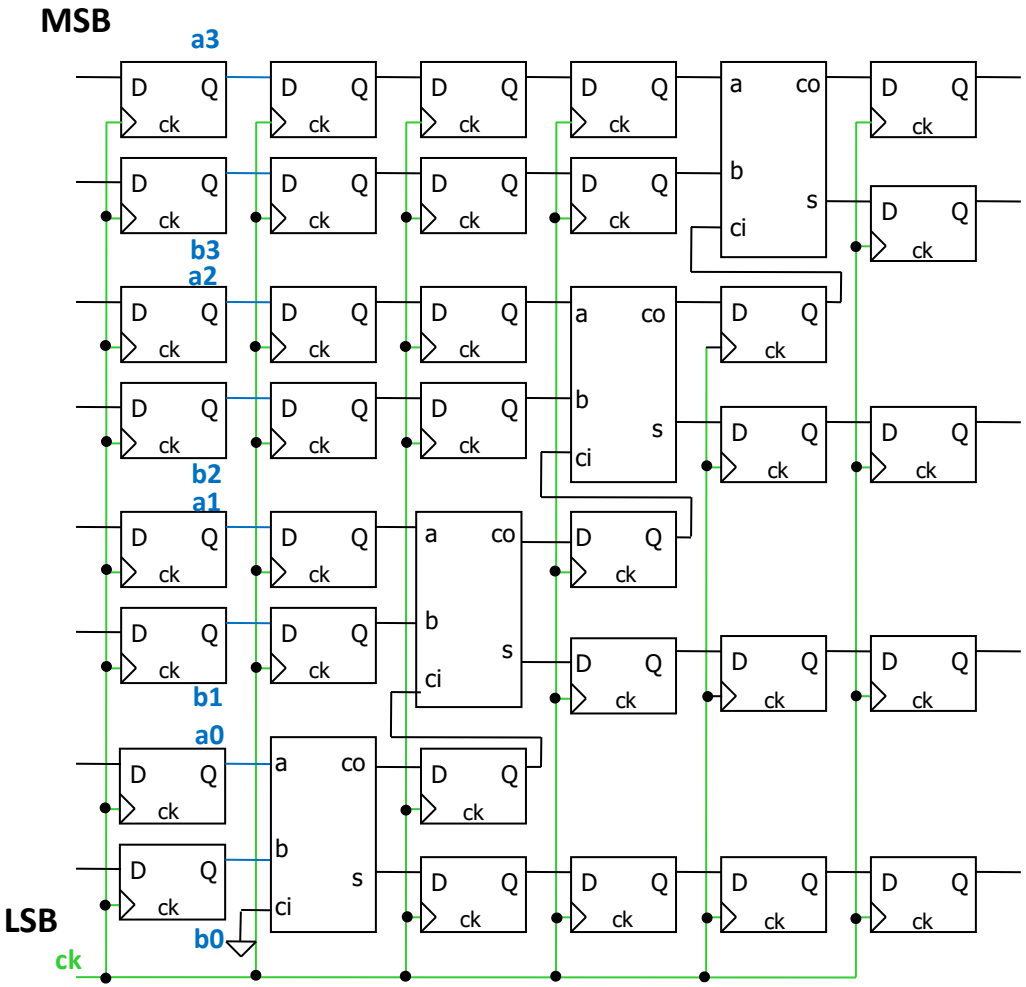
(a3 a2 a1 a0)=(0 1 1 1), (b3 b2 b1 b0)=(1 1 1 0) のとき
(s3 s2 s1 s0), (c3 c2 c1 c0) の値を求めよ



初期状態

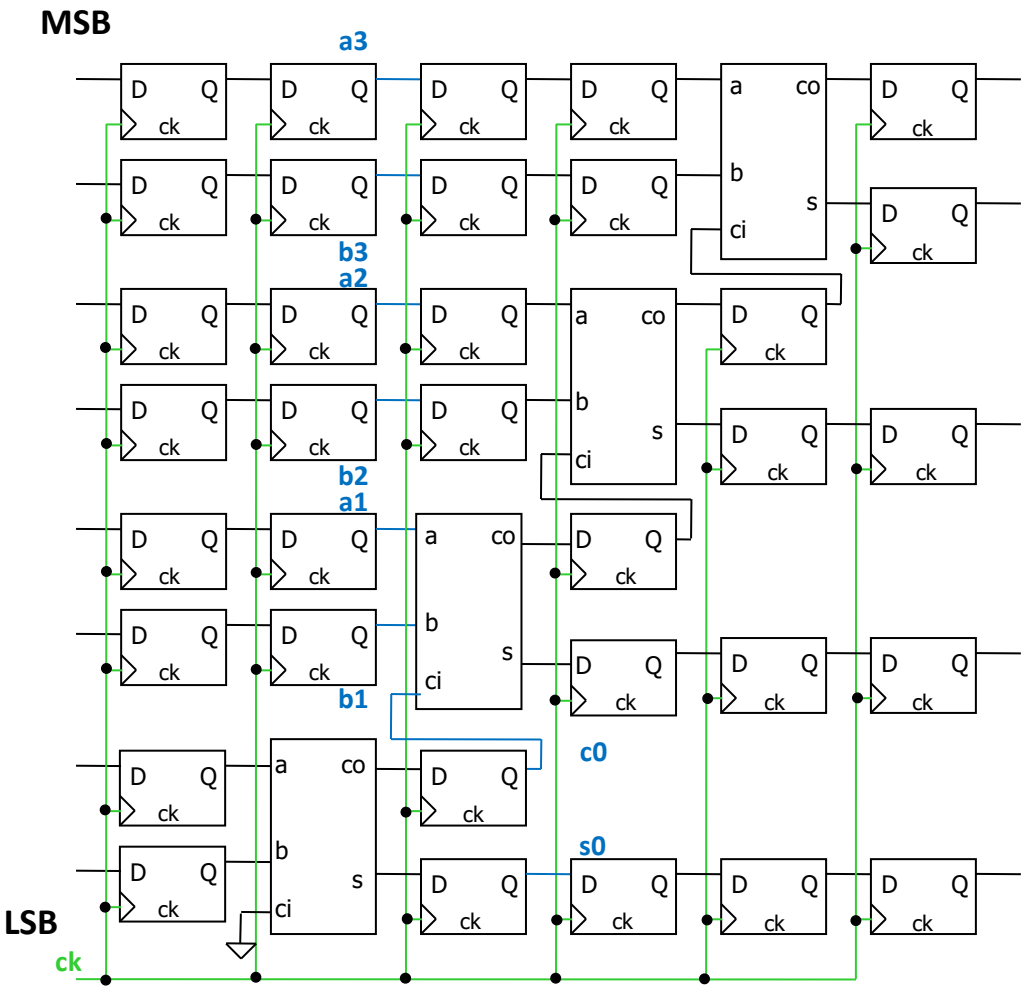
パイプライン加算器の動作

1クロック後



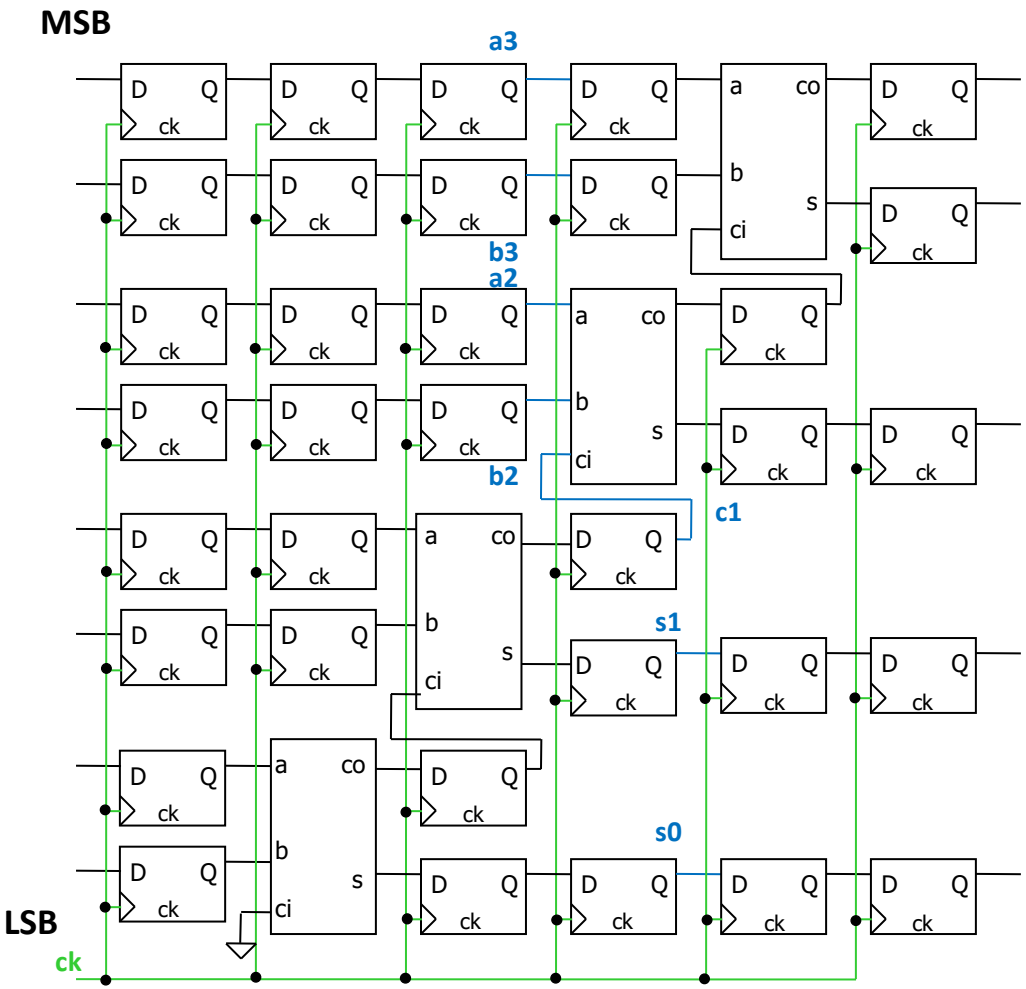
パイプライン加算器の動作

2クロック後



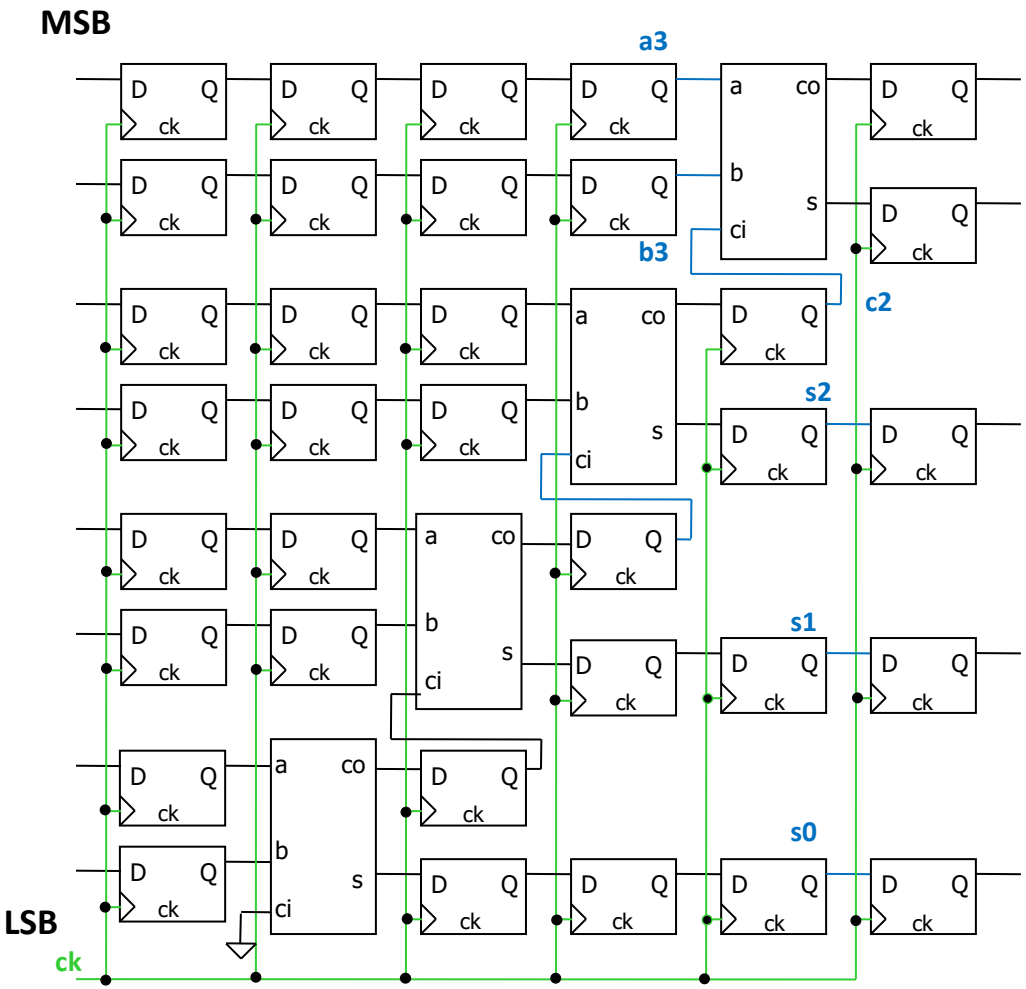
パイプライン加算器の動作

3クロック後



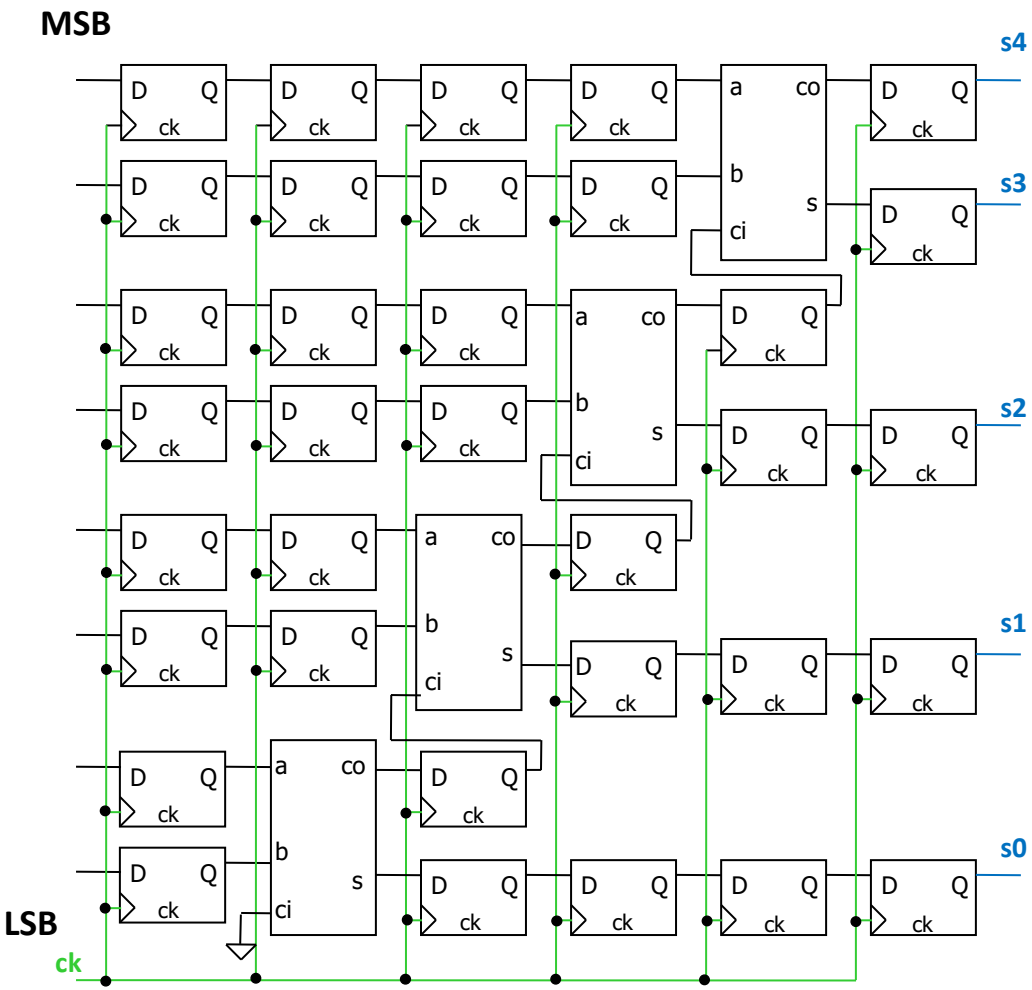
パイプライン加算器の動作

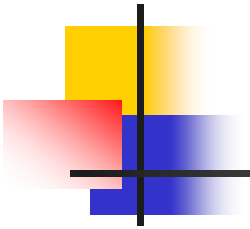
4クロック後



パイプライン加算器の動作

5クロック後





問題34 パイプライン加算器

パイプライン加算器の
メリット、デメリットを述べよ

問題35

ビットシフトで2^n 倍乗算を実現

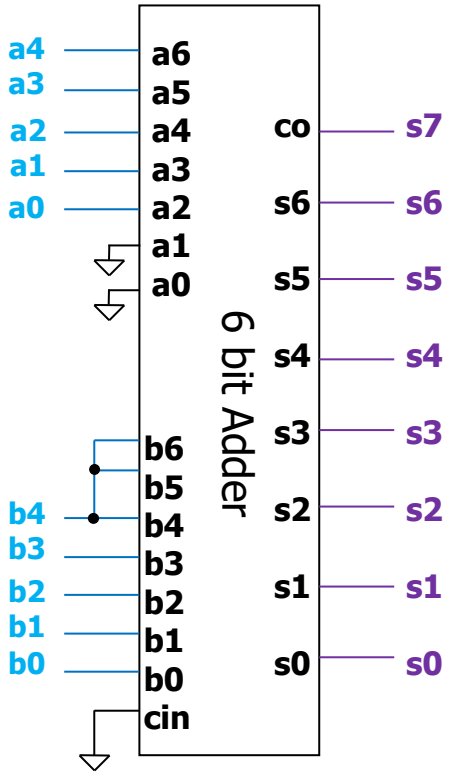
(a4 a3 a2 a1 a0)= (0 1 0 1 1)

(b4 b3 b2 b1 b0)= (0 1 1 1 1)

のとき、

(s7 s6 s5 s4 s3 s2 s1 s0)

を求めよ

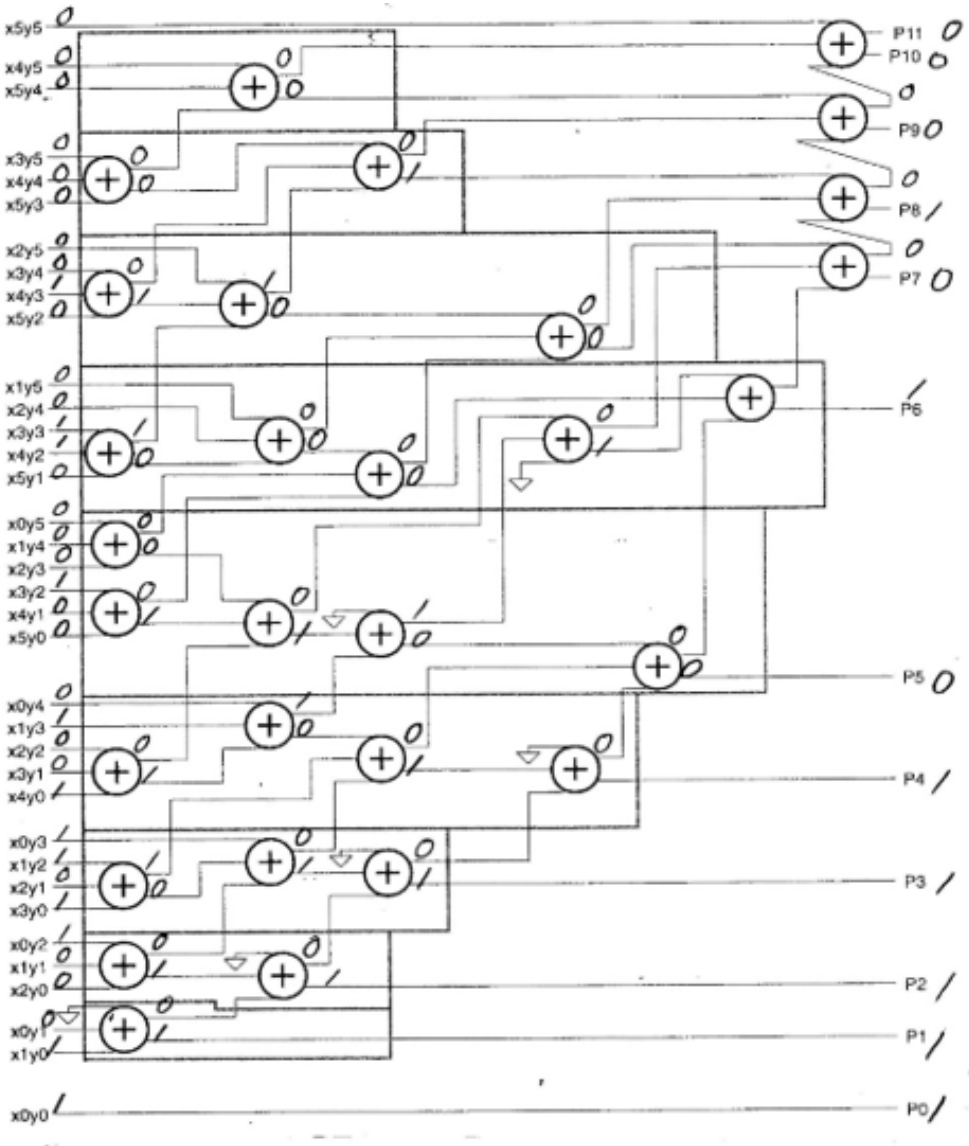


A, B, S の10進表現で

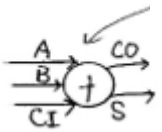
S= 4×A+B を確認せよ

問題36 デジタル乗算器

Wallace Tree Multiplier



Full adder



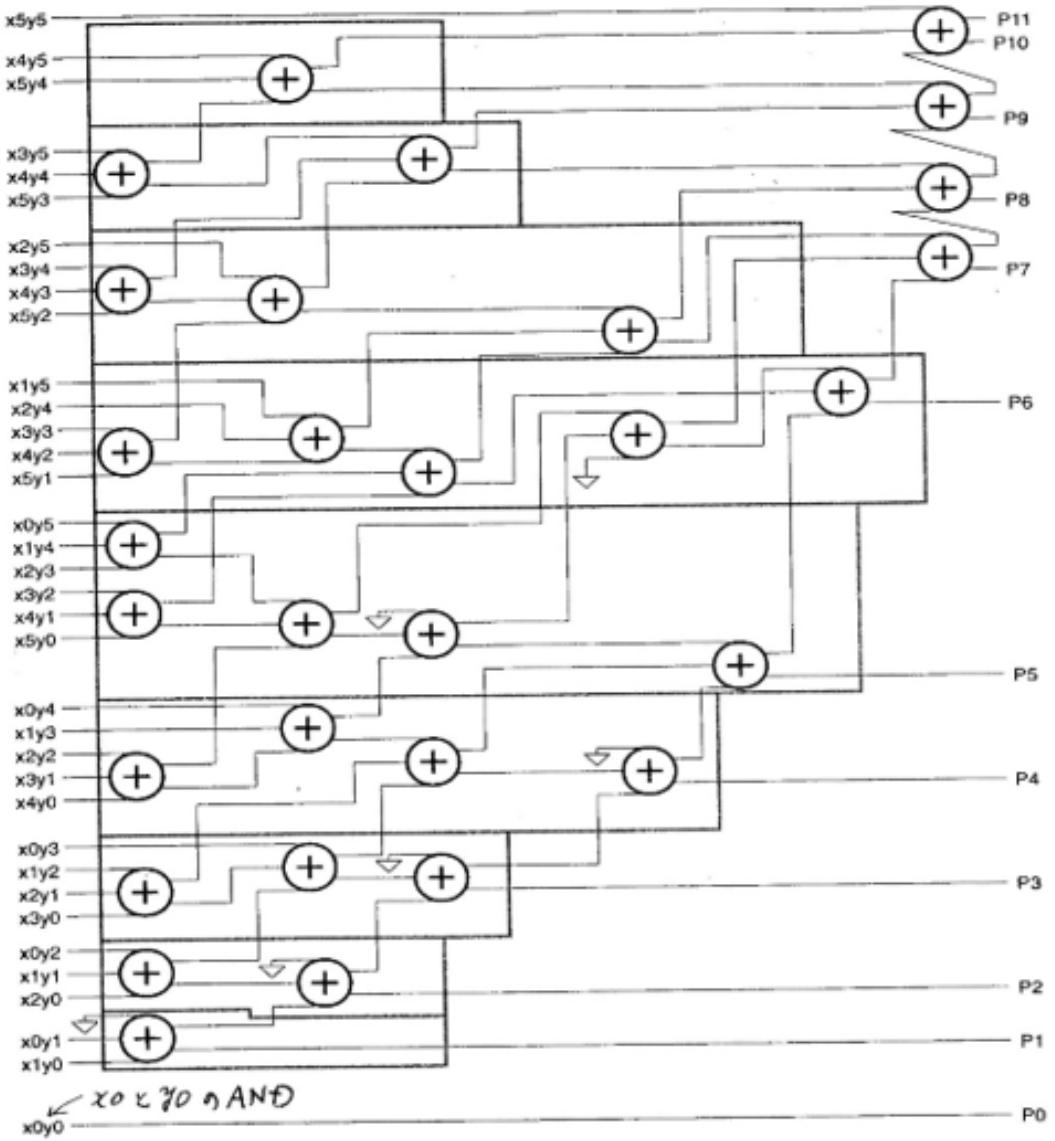
X 11011 10進で27
Y 01101 10進で13

P 101011111
10進で351

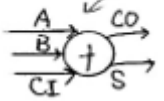
左図の回路は
P=XY
を実現している
ことを確認せよ

問題37 デジタル乗算器

Wallace Tree Multiplier



Full adder

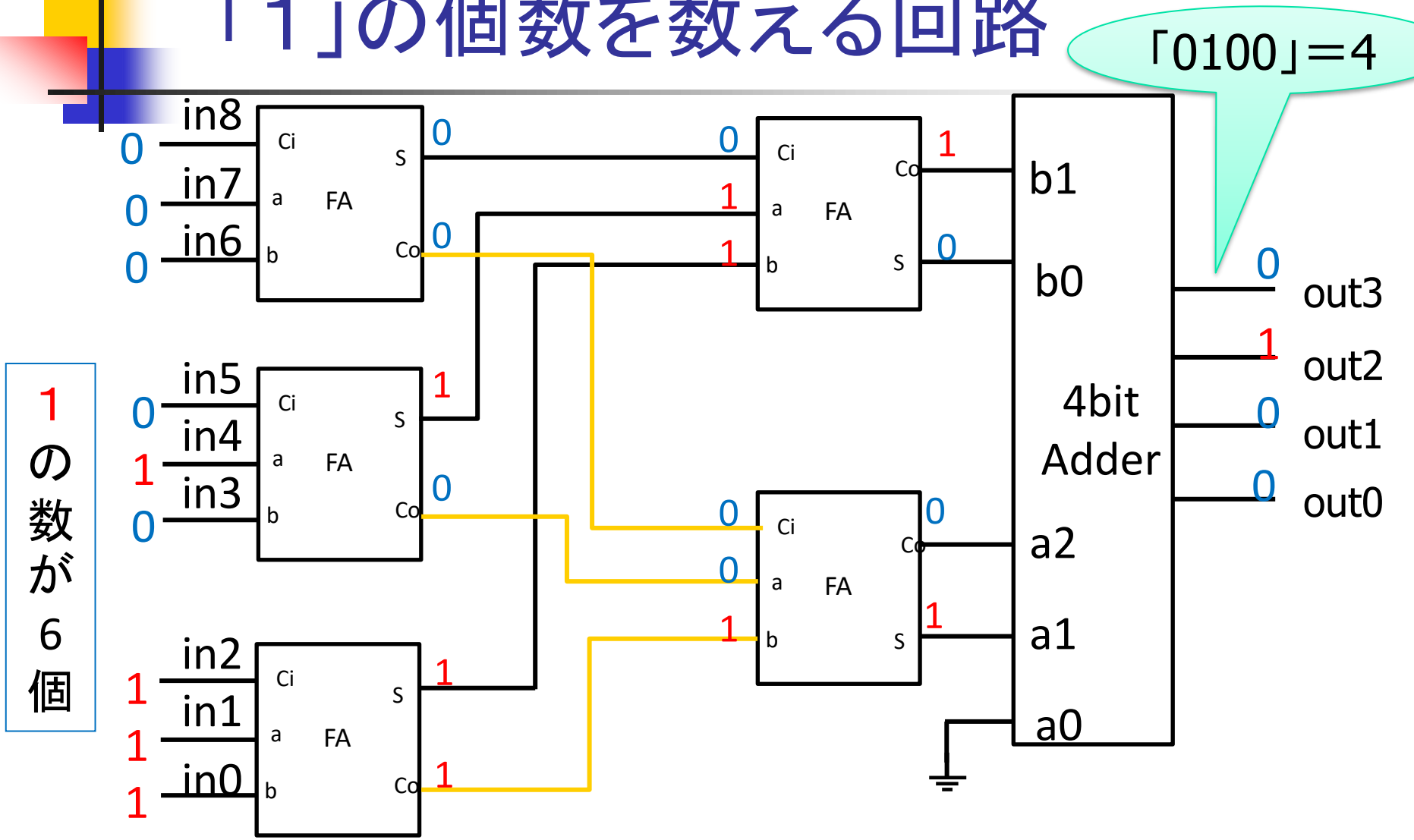


X 10110
Y 11101
のとき 左図の
各ノードの値(0 or 1)
を示し、Pを求めよ。

P=XYを確認せよ。

問題38 Carry Save Adder の応用

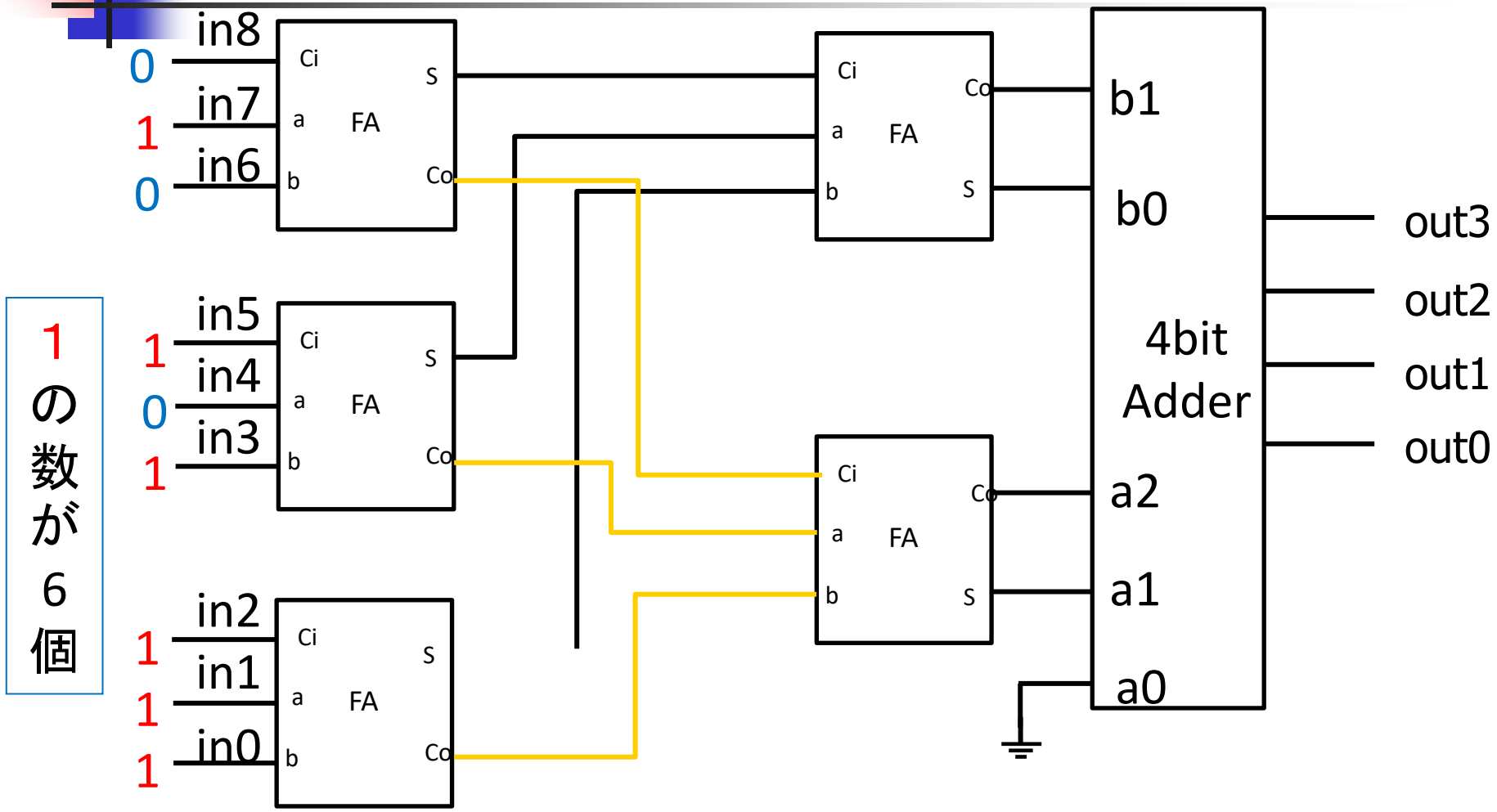
「1」の個数を数える回路



Carry Save Adder

問題38　Carry Save Adder の応用

前頁を参考に各ノードに0　or 1 を書け



Carry Save Adder

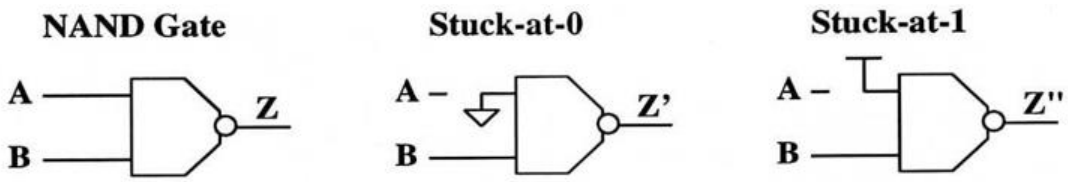
問題39　デジタルゲートの故障モデル 故障検出（テスト）

縮退故障モデル（Stuck-at fault model)

正常なNAND回路

入力が 0 に
はりつく故障

入力が 1 に
はりつく故障



A	B	Z	Z'	Z''
0	0	1	1	1
0	1	1	1	0
1	0	1	1	1
1	1	0	1	0

Stuck-at-1 is detected.

Stuck-at-0 is detected.

A=0, B=1 で検出可

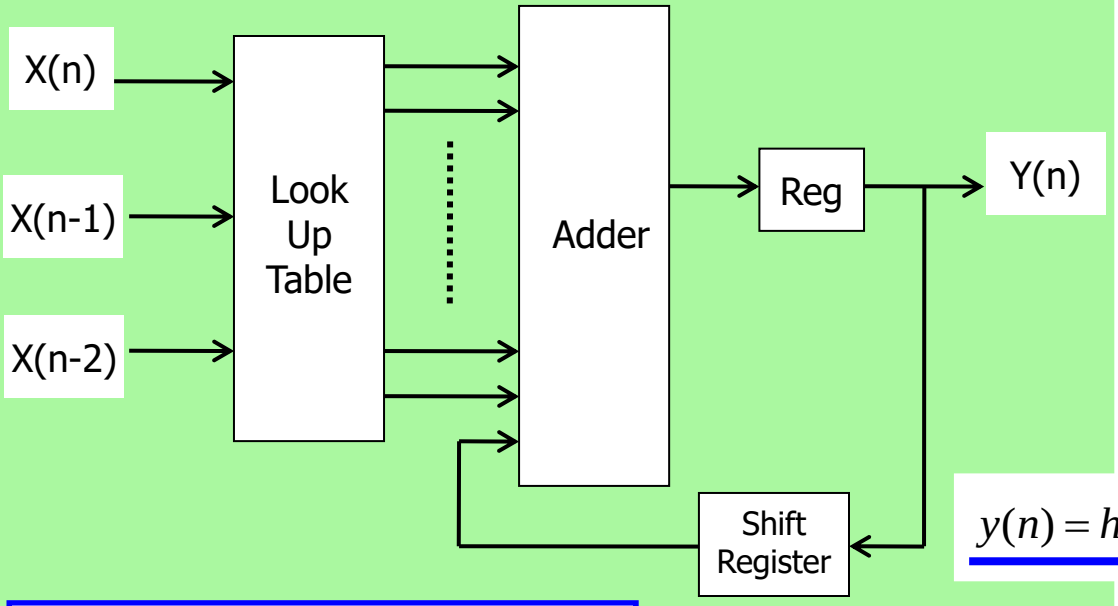
A=1, B=1 で検出可

問題：　入力A,BのNOR回路で　入力Aのstuck-at-0, stuck-at-1 をそれぞれ検出できる入力（A,B) は何か

問題40　乗算器を使用しない定係数積和演算回路

定係数の内積演算をルックアップ･テーブル（ROM等）により実現する計算手法

分散型積和演算



分散型積和演算回路構成

上位ビットからのシリアル演算

$$y(n) = h_0 x(n) + h_1 x(n-1) + h_2 x(n-2)$$

デジタル入力

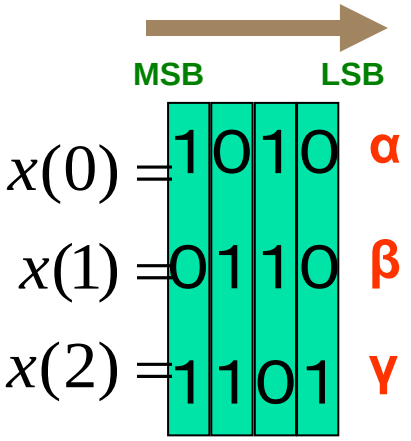
問題40-1

https://kobaweb.ei.st.gunma-u.ac.jp/news/pdf/2010/MWE2010HPslide.pdf
のp. 106 を参考のこと

分散型積和演算　動作原理

注意！

$$y(n) = h_0 x(n) + h_1 x(n-1) + h_2 x(n-2)$$



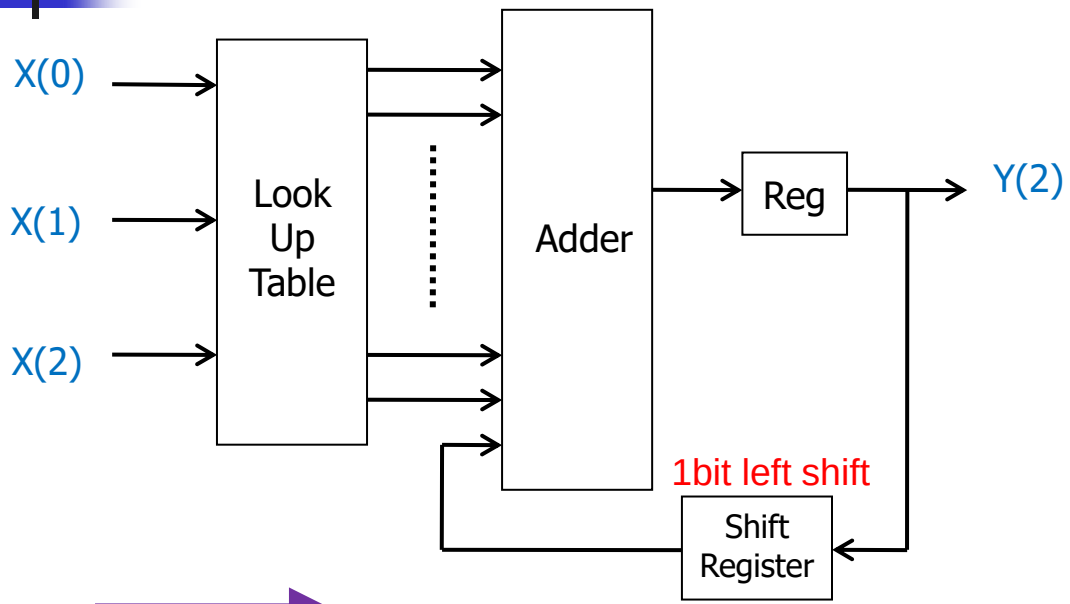
Look Up Table

ここをh0, h1, h2を用いて埋めよ

γ	β	α	Y(n)
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

問題40-2

分散型積和演算　動作原理



MSB　LSB

x(0)=1 0 1 0

x(1)=0 1 1 0

x(2)=1 1 0 1

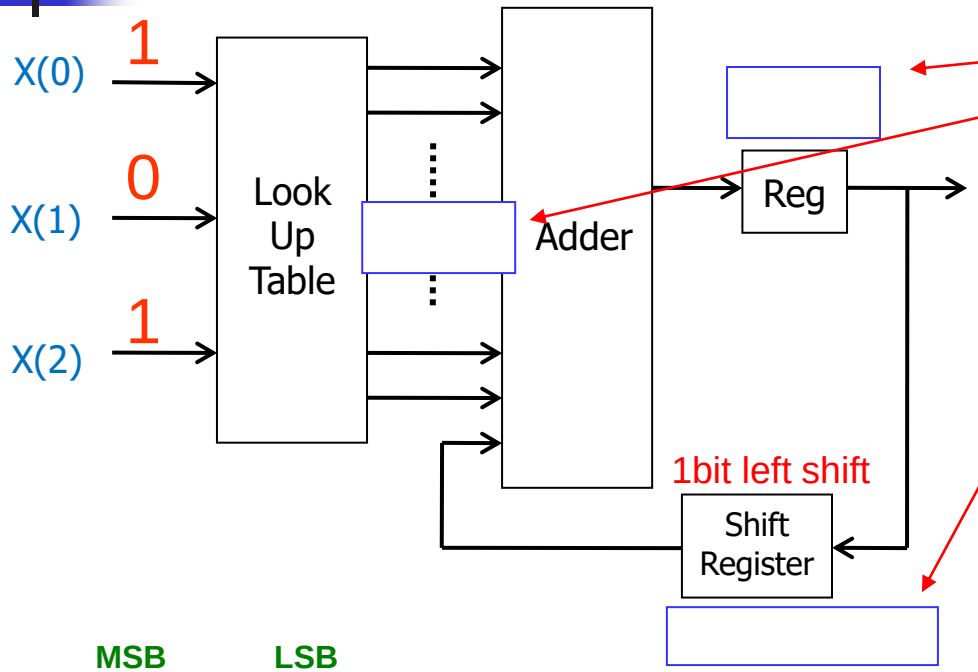
右シフトに処理

ここをh0, h1, h2を用いて埋めよ

γ	β	α	Y(n)
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

問題40-3

分散型積和演算　動作原理



ここをh0, h1, h2を用いて埋めよ

γ	β	α	Y(n)
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

MSB　LSB

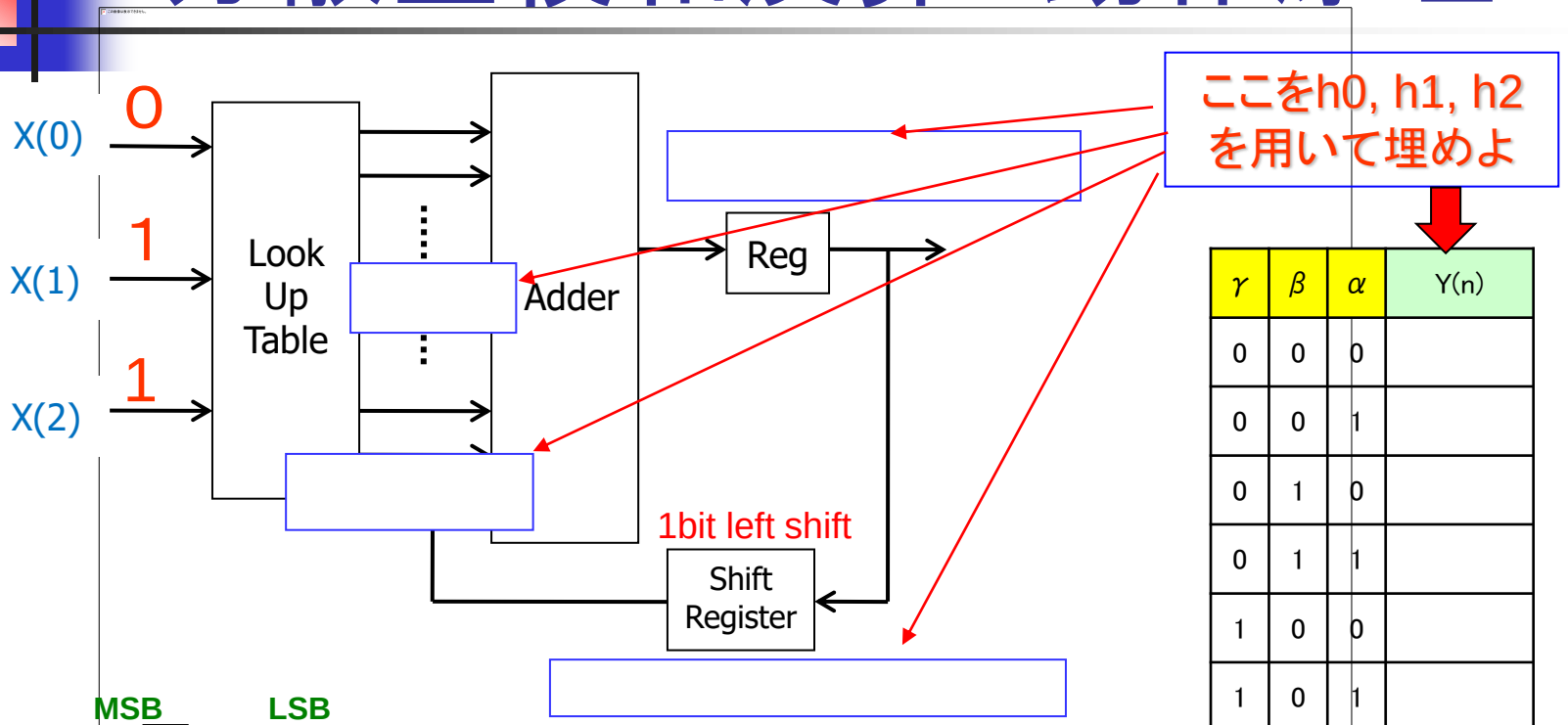
x(0)=1 0 1 0

x(1)=0 1 1 0

x(2)=1 1 0 1

問題40-4

分散型積和演算　動作原理



γ	β	α	Y(n)
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

MSB　LSB

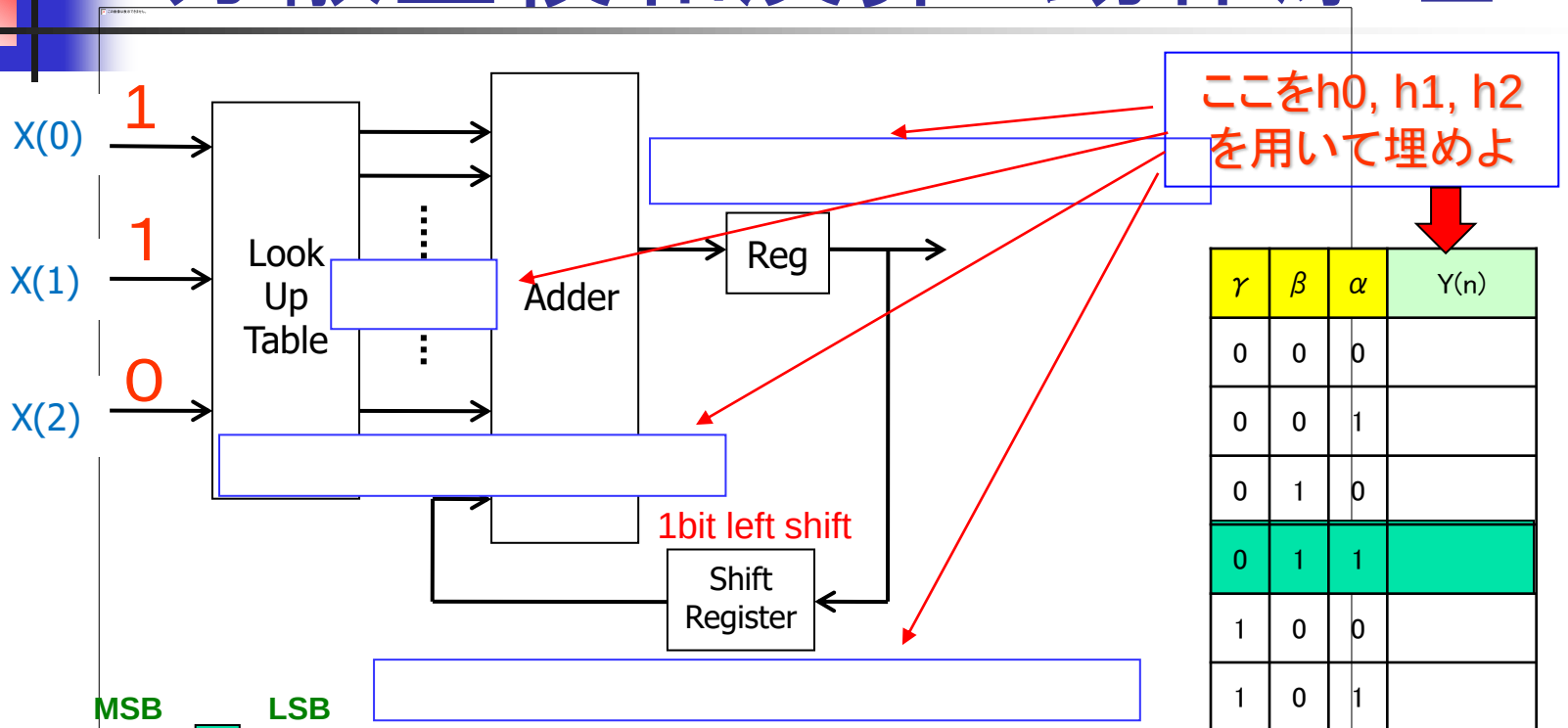
x(0)=1 0 1 0

x(1)=0 1 1 0

x(2)=1 1 0 1

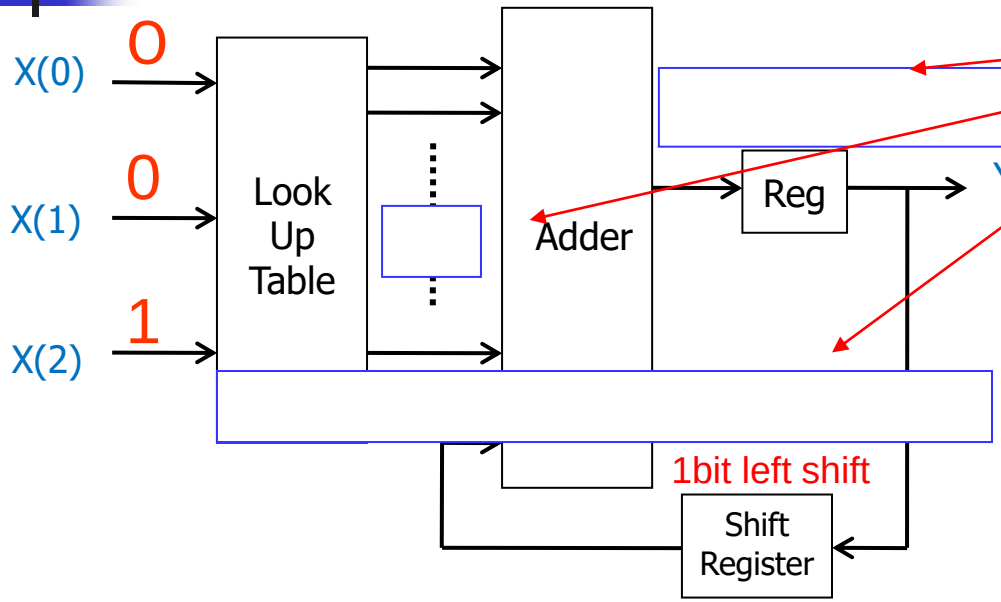
問題40-5

分散型積和演算　動作原理



γ	β	α	Y(n)
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

分散型積和演算　動作原理



ここをh0, h1, h2を用いて埋めよ

γ	β	α	Y(n)
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

MSB　LSB

x(0)=1 0 1 0

x(1)=0 1 1 0

x(2)=1 1 0 1

$Y(2)=h_0 X(0)+h_1 X(1) + h_2 X(2)$
になっていることを確認せよ

ディジタルアシスト・アナログ RF 技術のための ディジタル CMOS 回路設計再入門

Review of Digital CMOS Circuit Design for Digitally-Assisted Analog RF Technology

小林　春夫
Haruo KOBAYASHI
群馬大学大学院 工学研究科 電気電子工学専攻 〒376-8515 群馬県桐生市天神町 1-5-1
Graduate School of Engineering, Gunma University 1-5-1 Tenjin-cho, Kiryu, Gunma, 376-8515 Japan
Phone : 0277-30-1788 Fax :0277-30-1707 E-mail: k_haruo@el.gunma-u.ac.jp

Abstract
This article reviews digital CMOS circuit design for digitally-assisted analog RF technology, which is prevailing rapidly. Analog RF circuit designers have to learn digital CMOS circuit design for its implementation, and this article explains the followings:
(i)Transistor (switch) level CMOS digital circuit such as Inverter, NAND, NOR, Latch, Flip-Flop, complex logic gate
(ii) Digital CMOS circuit power consumption
(iii) Speed vs. supply voltage and temperature
(iv) Synchronous digital circuit design with examples of several counter circuits
(v) Several adder circuits for high speed
(vi) Distributed arithmetic circuit (multiply-add circuit with ROM and adder, and without multiplier)
(vii) Binary number theory for digital circuit design.

Keywords: Digital, CMOS, Power Consumption, Speed, Adder, Digitally-Assisted Analog RF Technology

1. はじめに

ディジタルアシスト・アナログ RF 技術が急速に普及する中[1,2]、アナログ RF 技術者も（比較的小規模の）ディジタル CMOS 回路、アルゴリズム設計を知らなければならない[3]。本稿では筆者がこの分野を理解するのに苦労したところ、面白いと感じたところ、勘違いしやすいところを踏まえてアナログ RF 回路設計者向けにディジタル CMOS 回路設計の基礎の説明を行なう。

2. ディジタルアシスト・アナログ RF 技術

LSI の微細化の進展とともに，ディジタル回路は面積の縮小・高速化・低消費電力化が進んでいる．しかし従来アナログ RF 回路では微細化に伴いトランジスタ特性ばらつき，真性利得低下，電源電圧低下のため必ずしも性能は向上せず，アナログ RF 回路設計のパラダイムシフトが必要である．

半導体プロセスの微細化はディジタルの低消費電力・高速・高集積化・低コスト化のために行う．したがってディジタルでメリットがなければ半導体微細化をする理由はない．微細化プロセスでもディジタルは必ず動作する．そこで微細 CMOS トランジスタを用いるLSIではディジタル技術を用いてアナログ性能を向上させる技術（ディジタルアシスト・アナログ RF 技術 Digitally-Assisted Analog RF Technology ）が重要な設計思想となり、急速に普及しつつある[1,2]。

別の側面から見れば、アナログ RF 技術者もそのためのディジタル CMOS 回路を、回路図を読んで理解し設計できなければならない。

3. PMOS, NMOS, CMOS スイッチ

PMOS, NMOS, CMOS スイッチの読み方:

CMOS LSI は PMOS トランジスタ、NMOS トランジスタの両方が使用できる LSI である。PMOS, NMOS とも Source (S), Drain (D), Gate (G) の3端子からなる。

PMOS スイッチはゲート(G)が Low（論理的に 0）のときスイッチオンで High (論理的に 1)のときスイッチオフである。

NMOS スイッチ は G=1 のときオン、G=0 のときオフとなる。

CMOS スイッチは PMOS と NMOS を並列に組み合わせたものである（図1）。

PMOS, NMOS, CMOS スイッチの特徴:

PMOS は G=0 のときオンであるが、ドレイン、ソース電圧が高いところでではオン抵抗が小さくなり、低いところではオン抵抗が大きくなる。

NMOS は G=1 のときオンであるが、ドレイン、ソース電圧が低いところでではオン抵抗が小さくなり、高いところではオン抵抗が大きくなる。

したがって PMOS は電源側、NMOS はグランド側で使用すると回路性能がよくなる。（図2）

CMOS スイッチはドレイン、ソース電圧が電源電圧からグランドの全範囲でオン抵抗が小さい。しかし回路規模、消費電力は大きくなる。

PMOS は G=0 でスイッチがオンでもドレイン電圧が 0 の場合はソース電圧は0にはならず|Vthp|までしか下がらない。

NMOS は G=1 でオンでもドレイン電圧が Vdd の場合はソース電圧は Vdd にはならず Vdd-Vthn までし

か上がらない。

CMOS スイッチの場合は出力は Vdd- 0 の間をフルスイングすることができる。ここで Vthp, Vthn は PMOS, NMOS のスレショルド電圧で、Vdd は電源電圧である。(図3)

4. ディジタル CMOS 回路(論理ゲート)

CMOS 回路で構成したインバータ回路を図4に、NAND 回路を図5に、NOR 回路を図6に示す。PMOS が電源側、NMOS がグランド側に使用されている。

複合論理の CMOS 回路例とその回路構成の規則を図7に示す。

論理積に対して PMOS は並列, NMOS は直列
論理和に対して PMOS は直列、NMOS は並列
最後に反転(インバータ)の記号をつける
の規則から構成される。

5. ディジタル CMOS 回路(メモリ素子)

ラッチ回路(Latch): ディジタル情報を記憶するラッチ回路はインバータ回路2つをリング状に接続して構成される。(図8)

フリップフロップ回路(Flip-Flop): ラッチ回路を2段接続し、後段では前段のクロックを反転したものを用いることでフリップフロップ回路が構成できる。(図9) フリップフロップ回路ではクロック CK の立ち上がりタイミングで入力データ D を取り込み保持する。

D は CK の立ち上がりのセットアップ時間前からホールド時間後まで確定していなければならない。(図10) セットアップ時間、ホールド時間がフリップフロップ回路設計者と使用者のインターフェース仕様になる。

高速ディジタル回路はほとんどの場合(ラッチではなく、回路量が2倍になるが)フリップフロップを用いる。

レジスタ回路(Register): クロックが共通のフリップフロップ配列である。

6. ディジタル CMOS 回路の消費電力

静的消費電力: 図4, 5, 6に示したディジタル CMOS 回路の動作からわかるように、Vdd からグランドへのスイッチはどこかで切れているので(わずかな漏れ電流を無視すれば)電流は流れていないので電力消費はゼロである(図 12)。これがディジタル CMOS 回路が低消費電力になり、LSI で CMOS が主流になった理由である。

温度が上昇するとスレショルド電圧が小さくなるため漏れ電流は増える。

動的消費電力: 実際には配線容量、次段ゲートへの入力容量(CL とモデル化する(図11))を介して、出力ノードが 0, 1 間をトグルする時に Vdd からグランドへ電荷が流れる。この動的消費電力 P は 負荷容量 CL のついた出力ノードが 1 秒間に f 回トグルすると

$$P = f C_L Vdd \times Vdd$$

となる。(図13) P は Vdd の2乗に比例するので電源電圧 Vdd を低くすると消費電力低減効果が大きい。

7. ディジタル CMOS 回路のスピード

電源電圧の影響: ディジタル CMOS 回路は電源電圧 Vdd を上げるとスピードが速くなる。

この理由は MOS の電流式を用いて説明できる。Vdd を上げると負荷容量 CL に蓄える電荷量 Q=CL Vdd は大きくなるが、MOS のドレイン電流がほぼゲート電圧(Vdd)の2乗に比例するため充放電の時間が小さくなるためである。(図14)

温度の影響: 温度が上昇すると電子および正孔の移動度が減少するためディジタル CMOS 回路のスピードは遅くなる。

8. マルチプロセッサ構成による低消費電力化

消費電力は Vdd の2乗に比例し、スピードは1乗に比例する。ディジタル回路の性能指標 (Figure of Merit : FOM) を スピード/消費電力とすると電源電圧 Vdd を下げるほどこの値が良くなる。

このことを利用し低電圧動作マルチプロセッサ構成により処理スピードを落とさずに低消費電力化する方式を図15に示す。微細 CMOS ではドレイン電流がゲート電圧の2乗ではなくγ乗に比例するので(1<γ<2)、この構成はさらに有効になる。

9.ディジタルの同期回路設計

ディジタル回路設計はタイミング設計を容易にするため基本的に同期回路設計を行う。(図16)

同期回路設計は図16に示すように全てのフリップフロップに同じクロック CK が入って動作する回路である。この CK の最小周期 T は

フリップフロップのセットアップ時間 ＋
内部組み合わせ回路の遅延 ＋
フリップフロップのホールド時間

である。これを満たすようにタイミング設計すればよい。

10.パイプライン構成による同期回路の高速化

パイプライン構成は CK の最小周期をさらに小さくする(最大周波数を高くする)ために、内部組み合わせ回路の中にレジスタを入れる方式である。(図17) 回路設計が容易のため広く用いられている。

11. 2進カウンタ回路

カウンタは文字通り(クロックパルスの)数を数える回路で、タイミング発生回路にも使用される。図18に2進カウンタのタイミングチャートを示す。Q0, Q1, Q2, Q3 出力はクロックが入力されると＋1 されるアップカウンタであり、その反転は-1 されるダウンカウンタである。

図19の上側は非同期カウンタ回路で下側が同期カウンタ回路である。このように非同期回路は小規模回路になることもあるが、設計・設計変更・デバッグ等が難しいので同期回路設計が基本である。

12. 様々なカウンタ

カウンタの変形として、**リングカウンタ**(図20)、**ジョ**

ンソンカウンタ（図21）、線形フィードバックシフトレジスタ（**Linear Feedback Shift Register: LFSR**）（図22）を示す。図20, 21, 22は同期回路である。

リングカウンタは逐次比較近似 ADC のタイミング発生回路等に用いられる。LFSR は疑似ランダム信号（M 系列信号）を発生し LSI テスト用の Built-In Self-Test 回路の一部等で使用される。

13. ディジタル加算器

ディジタルは2進数で演算されるが、2進数の加算アルゴリズムとその基本回路の全加算器（**Full Adder**）の真理値表を図23に示す。

Full Adder を4つ用いた4ビットの加算器（**Carry Ripple Adder**）を図24に示す。桁上げの伝播（Carry Propagation）が加算器のスピードを律則するのでその問題を軽減するために様々な構成が考案されている。

その一つとして図25右に8ビットの桁上げ選択加算器(**Carry Select Adder**)を示す。上位4ビットに対して、下位4ビットからの桁上げが0の場合と1の場合の両方を計算する。下位4ビットからの桁上げが計算されたときその0, 1の値に応じてマルチプレクサで上位4ビットの値を選択する。

図26に**パイプライン加算器**を示す。図17のパイプライン回路の原理に従い、最高クロック周波数が高速になり、すなわち **throughput** が高くなり開ループシステムでは有効である。しかしある入力データに対応する結果が計算されるまでのクロック数が5となり、すなわち遅延（**latency**）が大きくなり、入力後の出力がすぐ必要な閉ループシステム等ではこの構成は使えないこともあるので注意が必要である。

（2入力ではなく）多入力の加算（Z=A+B+C+D+…）で桁上げ伝播をできるだけ少なくする方式を考える。直接的な方式では2入力加算器をいくつも使用する方式である（ Z= ((A+B) + (C+D))+…)。これに対して桁上げ節約加算器(**Carry Save Adder**) は最後に1回だけ桁上げ演算を行うので高速、回路規模小になる。図24に3入力加算の場合の Carry Save Adder を示す（S=X+Y+Z）。

別の観点からは、回路規模を小さくする**ビットシリアル加算器**もある。また、桁上げ伝播時間を小さくする他の良く知られた方式して **Carry Look-ahead Adder, Conditional Sum Adder** 等もよく知られている。

14. ビットシフト、ディジタル乗算器

10進数で x10, x100, x1000, ÷10, ÷100 等は小数点の位置をずらすだけでよいので簡単に計算できる。同様に2進数表現では x2, x4, x8, ÷2, ÷4 等は小数点の位置をずらす（ビットシフト）だけでよいので簡単に計算できる。（図28）

ビットシフトは浮動小数点の加算の演算で2つの入力指数部を合わせるところでも使用される。

1クロックで何ビットもビットシフトできる回路 **Barrel Shifter** はマルチプレクサ配列等で実現できる [3].

乗算は加算を何回も繰り返すことで計算できるので（図30）ディジタル乗算器は全加算器の 2 次元配列になる。加算器で高速化、回路小規模化の工夫が様々なされてきているように、乗算器でも **Wallace Tree Multiplier, Booth Algorithm Multiplier** 等さらに工夫がなされている。

いずれにせよディジタル乗算器は回路規模が大きいので、ビットシフトを利用して乗算器を用いないほうがよい。例えば図29に S= 4xA+B の乗算器を用いずビットシフトを使用した回路構成を示す。

15. 分散型定係数積和演算回路

定数 h0, h1, h2… に対して 入力 x(n) の積和演算

$$y = h0\ x(n) + h1\ x(n-1) + h2\ x(n-2) + \cdots$$

を乗算器を用いずに 定数を記憶する ROM と加算器で構成する方式を図 31, 32 に示す。乗算器を用いた場合に比べて、回路規模が小さくなるだけでなく計算スピードも同等である。ビットシリアル演算のアルゴリズムを用いており、古くから広く産業界でも応用されている。アルゴリズム、回路実現とも面白い方式である。

16. まとめ

近年普及が著しいディジタルアシスト・アナログ RF 技術でのディジタル演算（おもに加算と乗算からなる）を高速で低消費電力で実行するディジタル CMOS 回路の設計について、容易に理解できるように筆者の経験に基づきポイントの説明を行った。

付録に 2 進数、8 進数、16進数、負の数の2の補数による表現、なぜ2の補数表現を用いるのかの考え方を示した。数論の回路設計へのさらなる展開として、非 2 進数を SAR ADC に応用することも行われている [4]。

謝辞: 本講座の機会を与えていただいた田中聡氏、原稿の図の作成を支援してもらった孫清波君に感謝いたします。

参考文献

[1] 小林 春夫「ナノCMOS時代のアナログ回路 -ディジタルアシストAD変換技術を中心として-」電子情報通信学会、第22回 回路とシステム（軽井沢）ワークショップ（2009年4月）.

[2] R. B. Staszewski, P. T. Balsara, *All-Digital Frequency Synthesizer in Deep-Submicron CMOS* Wiley-Interscience (1996)

[3] N. H. E. Weste, K. Eshraghian, *Principles of CMOS VLSI Design – A System Perspective* -, Addison Wesley; Second edition (1994).

[4] T. Ogawa, H. Kobayashi, Y. Takahashi, N. Takai, M. Hotta, H. San, T. Matsuura, A. Abe, K. Yagi, T. Mori, “SAR ADC Algorithm with Redundancy and Digital Error Correction”, IEICE Trans. Fundamentals, vol.E93-A, no.2, pp.415-423 (Feb. 2010).

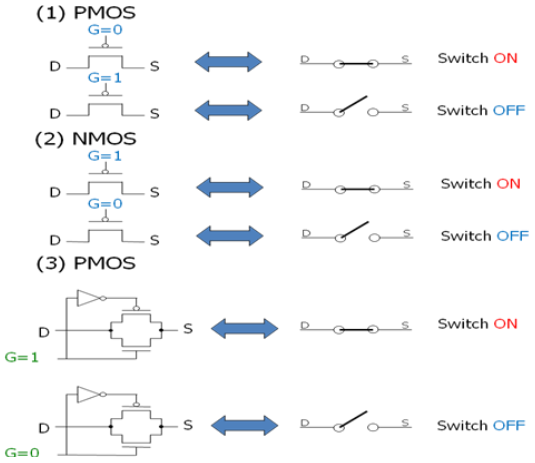


図 1　PMOS, NMOS, CMOS スイッチ

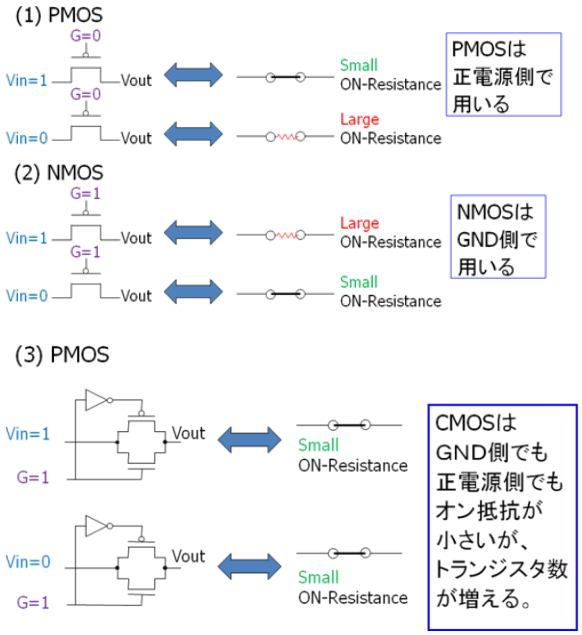


図 2　PMOS, NMOS, CMOS スイッチのオン抵抗

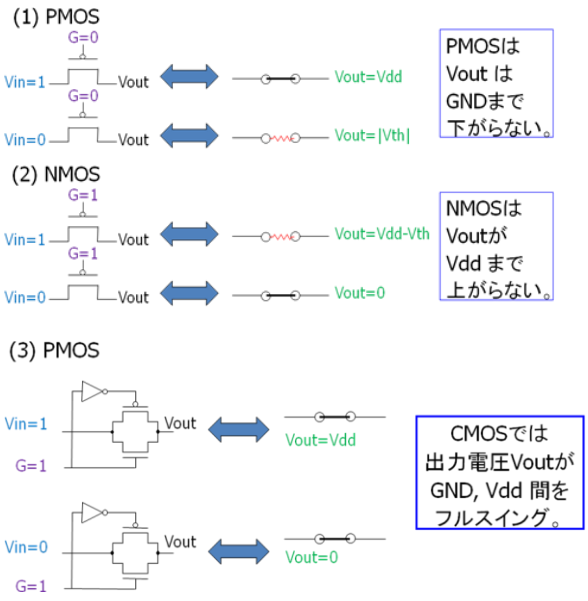


図 3 PMOS,NMOS,CMOS スイッチの出力電圧

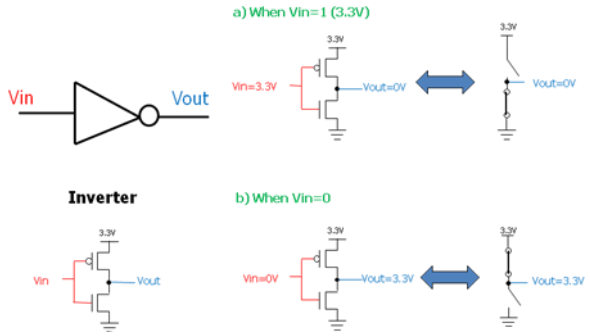


図 4　CMOS Inverter 回路

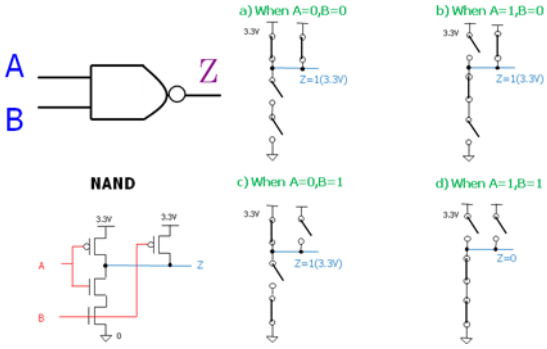


図 5　CMOS NAND 回路

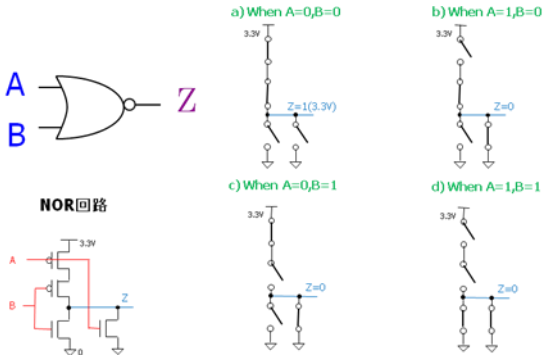


図 6　CMOS NOR 回路

	論理積	論理和
PMOS	並列	直列
NMOS	直列	並列

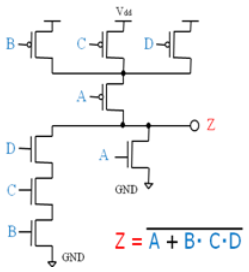


図 7　CMOS 複合ゲートの回路構成の規則と回路例

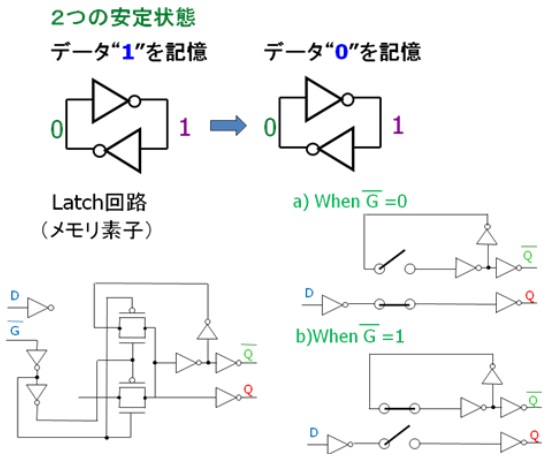


図 8　CMOS ラッチ回路

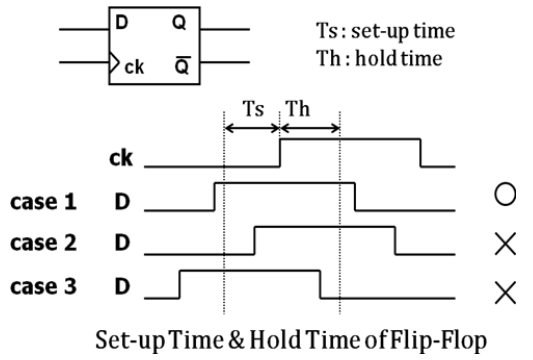


図 9　Flip-Flop 回路インターフェース仕様

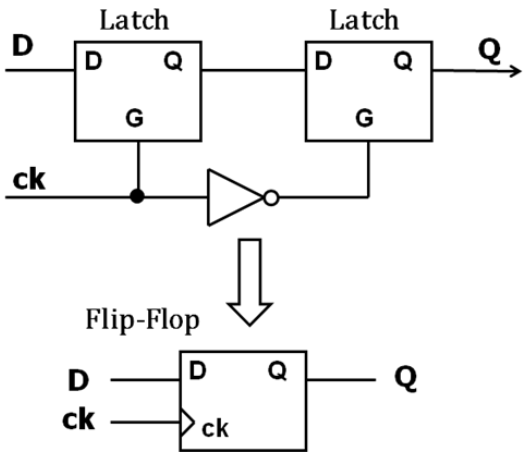


図 10　Flip-Flop 回路の構成

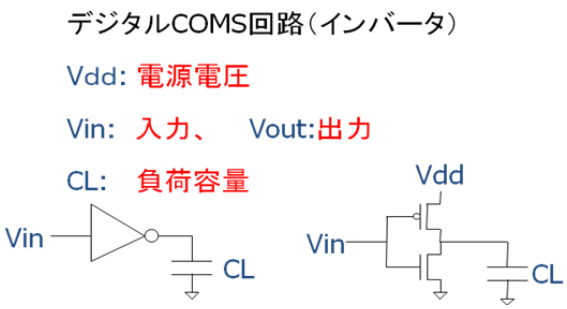


図 11 負荷容量 CL

（注）最近の微細CMOSデジタル回路では リーク電流
が大きくなり、静的電力消費の占める割合が増えてきている。

図 12　CMOS インバータ静的消費電力

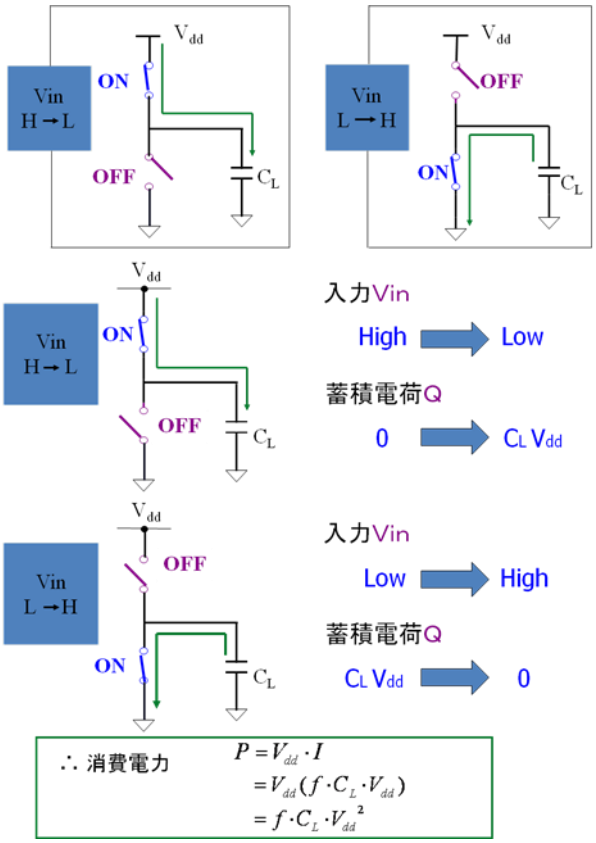


図 13　CMOS インバータ動的消費電力

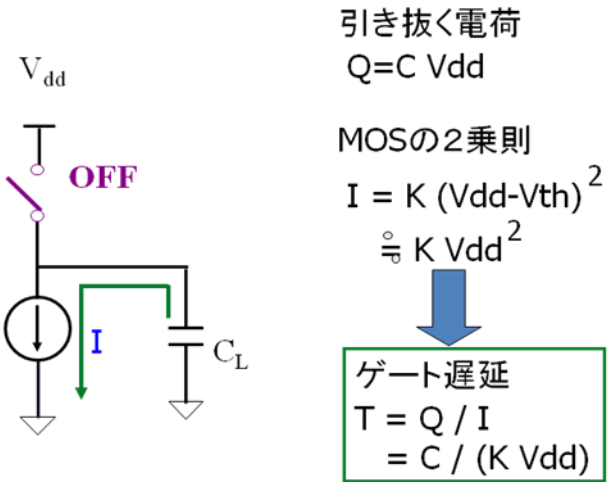


図 14　CMOS インバータの遅延（スピード）

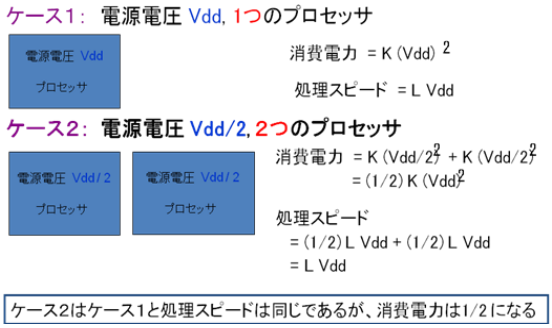


図 15　マルチプロセッサ構成による低消費電力化

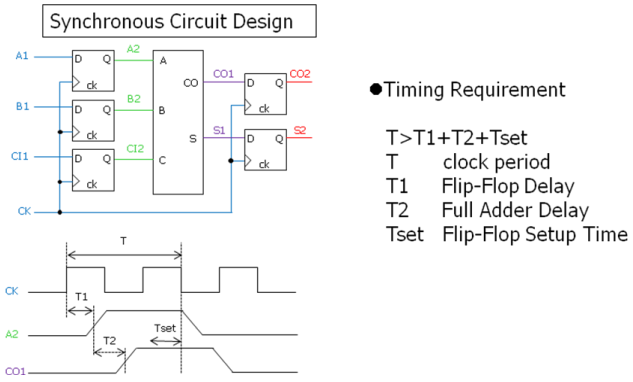


図 16　同期回路とタイミング設計

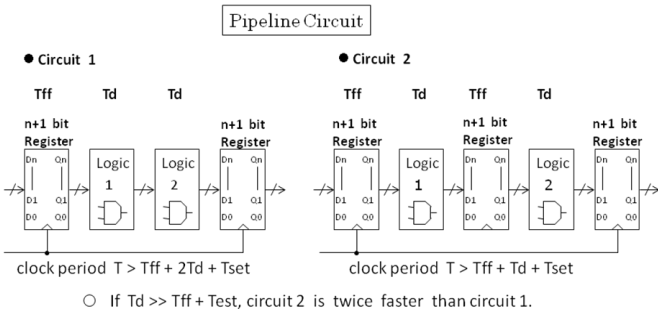


図 17　パイプライン回路

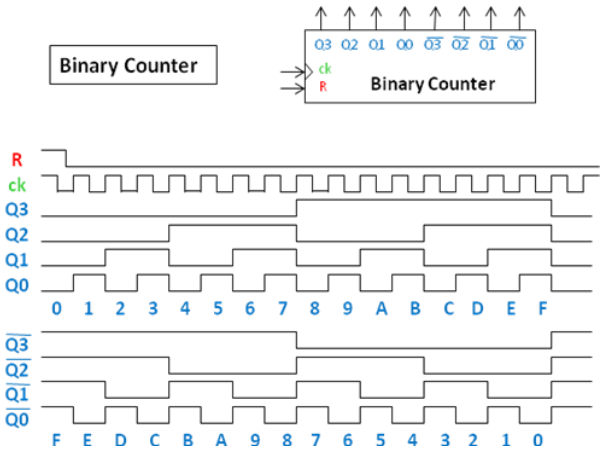


図 18　2 進カウンタとタイミングチャート

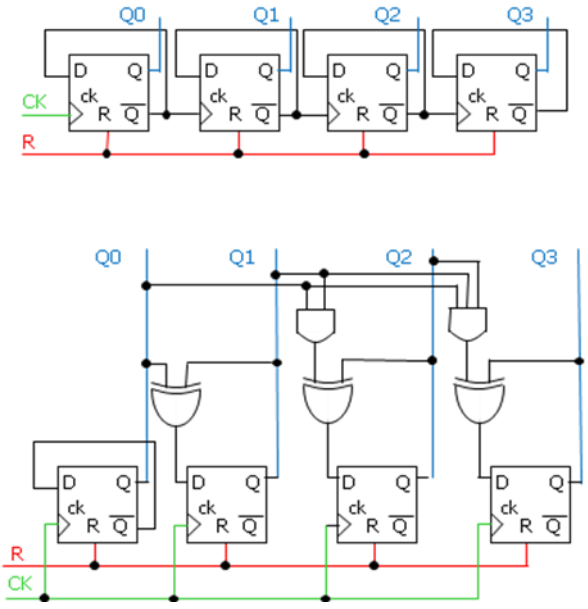


図 19　（上）非同期、（下）同期カウンタ回路

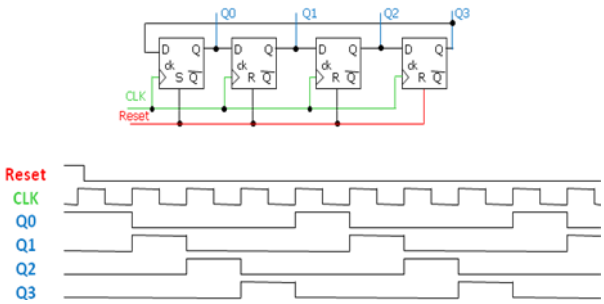


図 20　リングカウンタ

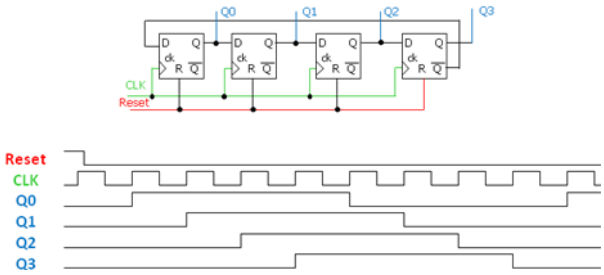


図 21　ジョンソンカウンタ

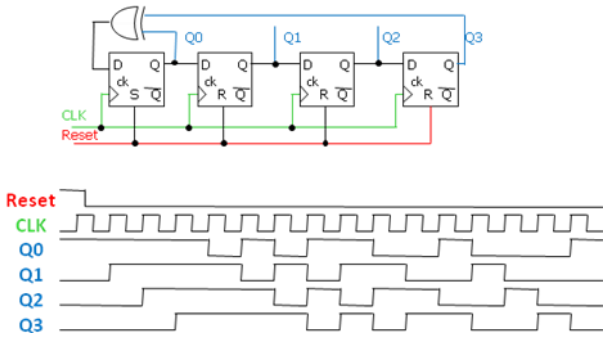


図 22　M 系列発生回路（LFSR）

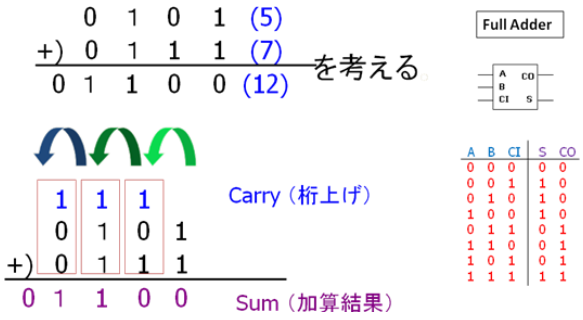


図 23　２進数加算アルゴリズムと Full Adder

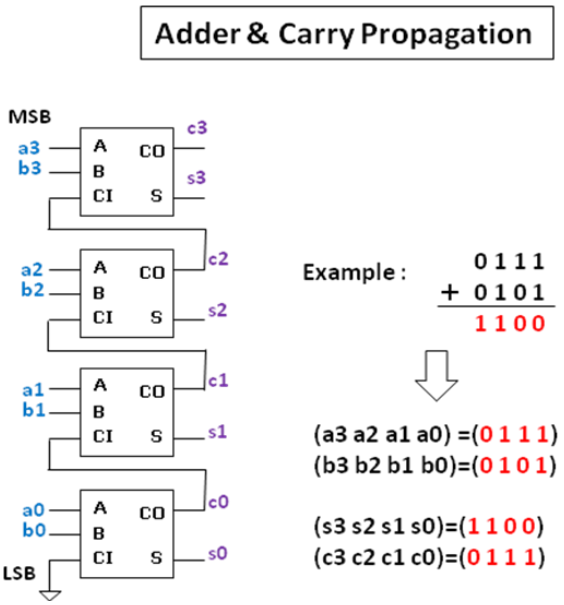


図 24　4bit Carry Ripple Adder

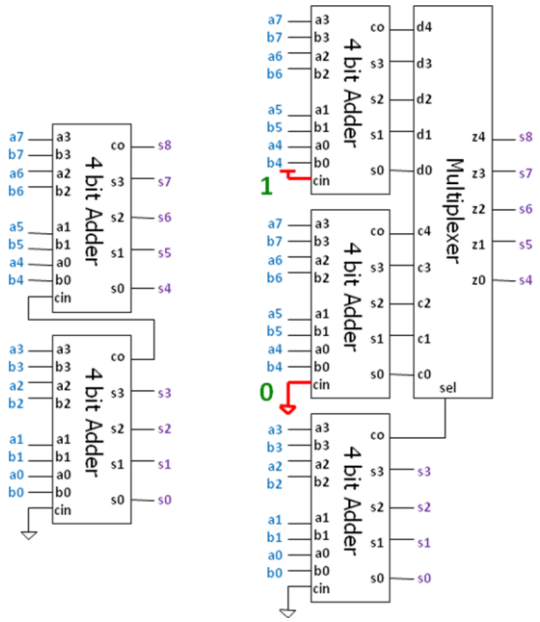


図 25　8bit Adder　（左）通常構成

（右）8bit Carry Select Adder

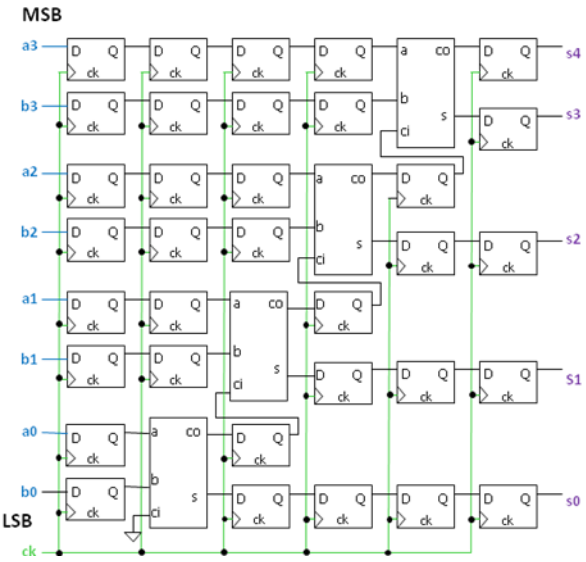


図 26 (a)　パイプライン加算器の構成

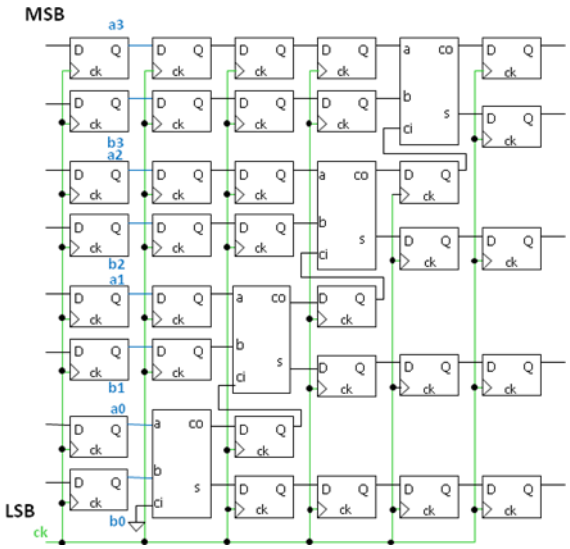


図 26 (b)　パイプライン加算器　１クロック目

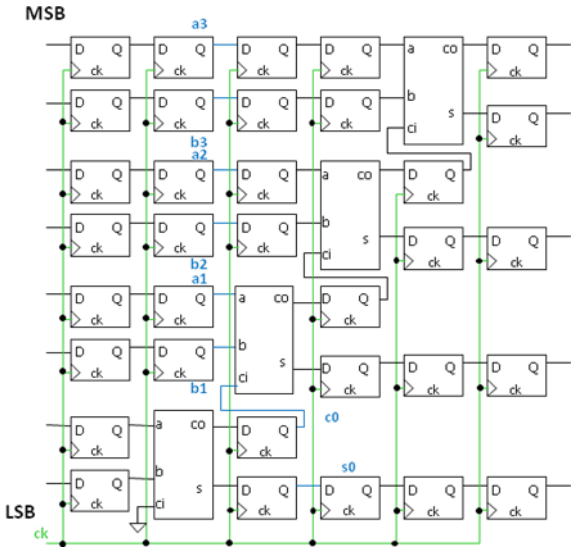


図 26 (c)　パイプライン加算器　２クロック目

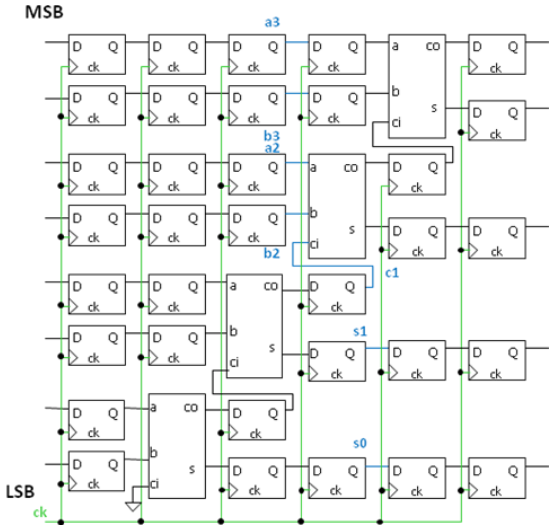


図 26 (d) パイプライン加算器　3 クロック目

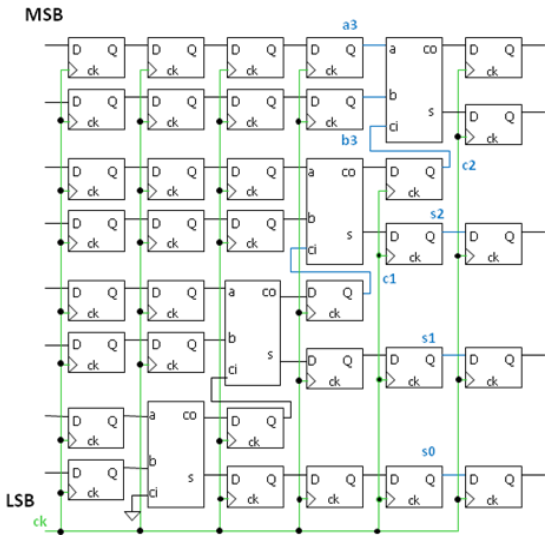


図 26 (e) パイプライン加算器　4 クロック目

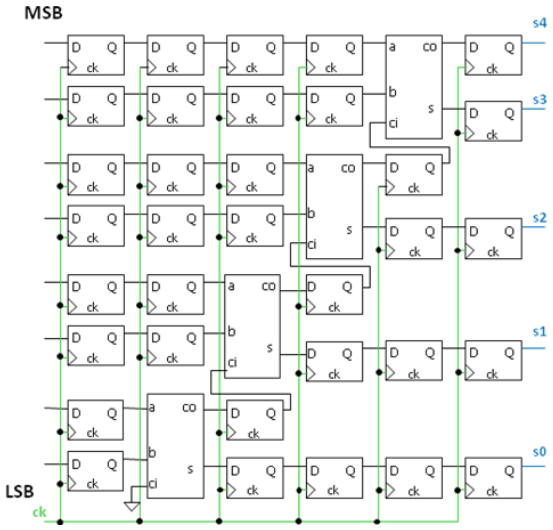


図 26 (f) パイプライン加算器　5 クロック目

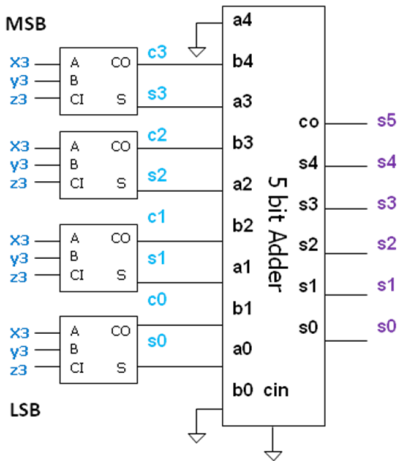


図 27　Carry Save Adder (S=X+Y+Z)

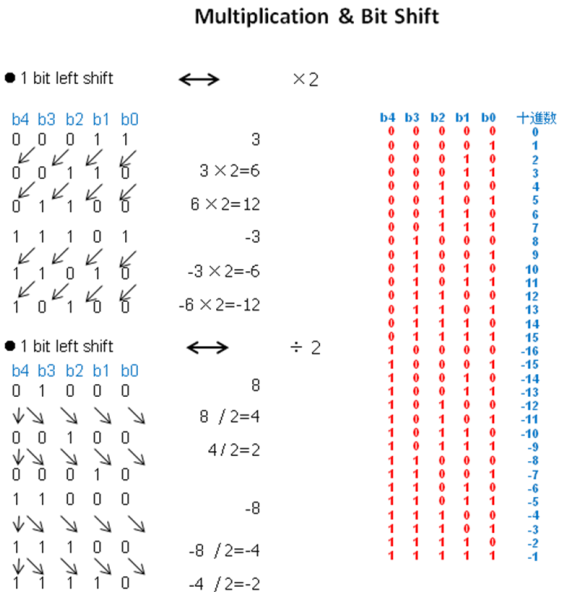


図 28　ビットシフト

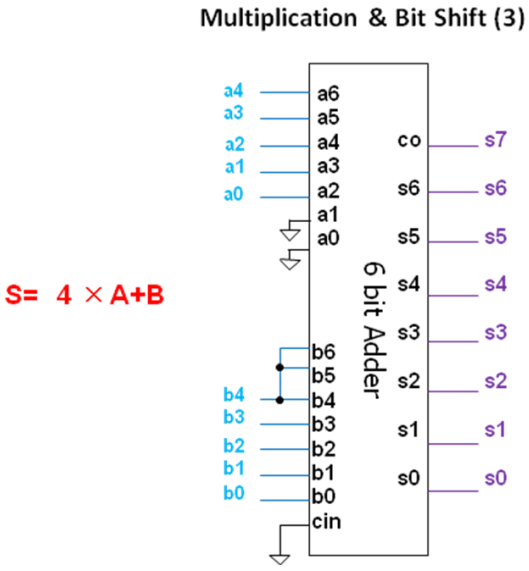


図 29　ビットシフトと加算

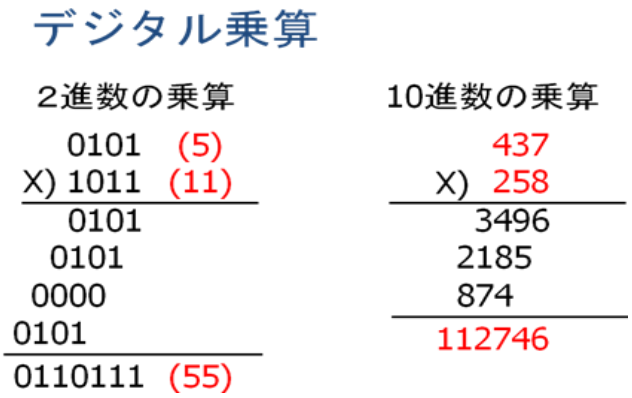


図 30　乗算アルゴリズム

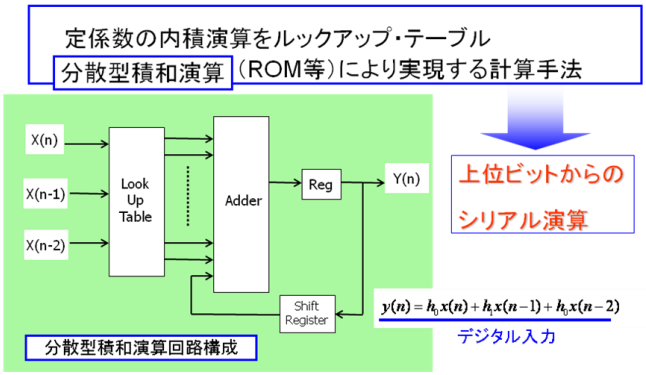


図 31　分散型積和演算器構成

$$y(n)=h_0x(n)+h_1x(n-1)+h_0x(n-2)$$

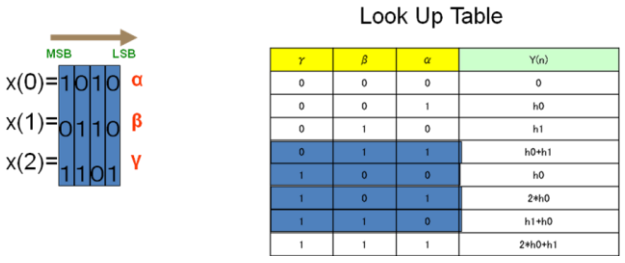


図 32　ROM 内部の説明

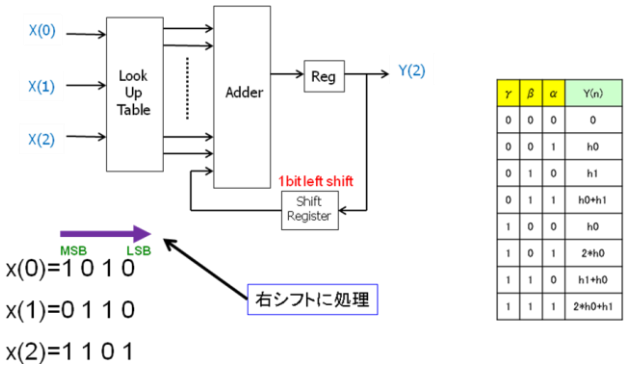


図 33 (a)　分散型積和演算器動作

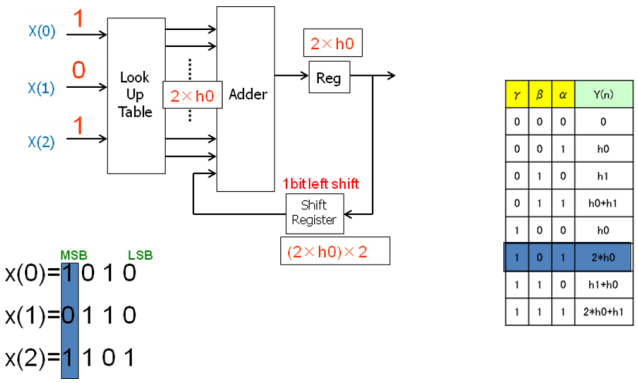


図 33 (b)　分散型積和演算器動作 1

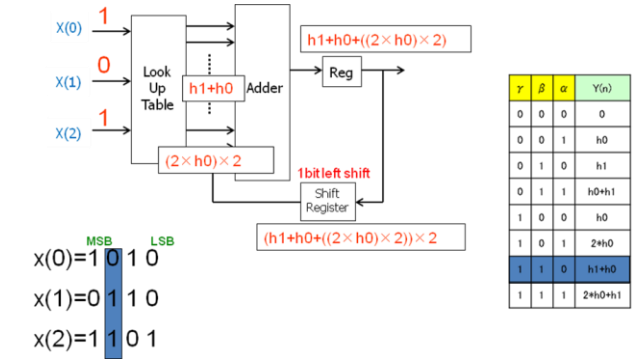


図 33 (c)　分散型積和演算器動作 2

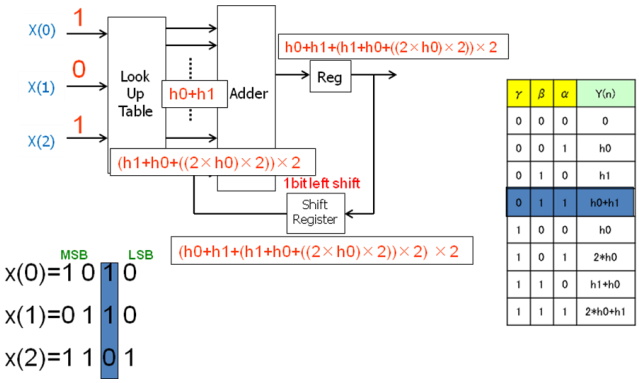


図 33 (d)　分散型積和演算器動作 3

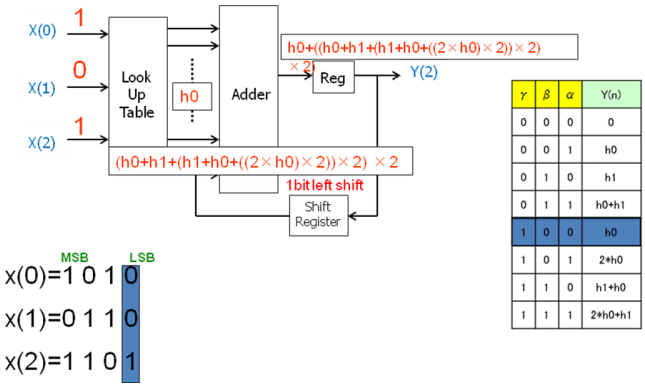


図 33 (e)　分散型積和演算器動作 4

附　録

デジタル回路と２進数

- 人間はなぜ１０進数を使うか？
  → 手の指が１０本あるから。
- デジタルではなぜ２進数を使うか？
  → ２つの状態は技術的に容易かつ安定して実現可能。

例：　電圧の高いと低い
　　電流の流れる状態と流れない状態
　　パルスのあるとなし。

１６進数、８進数とデジタル

10進　0 1 2 3 4 5 6 7　8 9 10 11 12 13 14 15 16 17 18 19 20
8進　0 1 2 3 4 5 6 7 10 11 12 13 14 15 16 17 20 21 22 23 24
16進 0 1 2 3 4 5 6 7　8 9　A　B　C　D　E　F 10 11 12 13 14

- 人間はなぜ１０進数を使うか？
  → 手の指が１０本あるから。
- デジタルコンピュータは２進数が基本。
  ではなぜ１６進数、８進数を使うか？
  → ２進数と１６進数、８進数は相性がよいから。

８進数と２進数の変換

8進	2進 (4 2 1)
0	0 0 0
1	0 0 1
2	0 1 0
3	0 1 1
4	1 0 0
5	1 0 1
6	1 1 0
7	1 1 1

例　8進4桁　3724
- ●10進に変換
  3x8x8x8+7x8x8+2x8+4
  計算が必要
- ●2進に変換
  011 111 010 100
  左表から機械的に得られる

１６進数と２進数の変換

16進	2進	16進	2進
0	0000	8	1000
1	0001	9	1001
2	0010	A	1010
3	0011	B	1011
4	0100	C	1100
5	0101	D	1101
6	0110	E	1110
7	0111	F	1111

例　16進で3桁
　A46
2進数に変換
1010 0100 0110
左表から機械的に得られる

負の数の表現（２の補数）

4ビットの場合

符号無 ud	符号付 sd	2進 b3b2b1b0	符号無 ud	符号付 sd	2進 b3b2b1b0
0	0	0000	8	-8	1000
1	1	0001	9	-7	1001
2	2	0010	10	-6	1010
3	3	0011	11	-5	1011
4	4	0100	12	-4	1100
5	5	0101	13	-3	1101
6	6	0110	14	-2	1110
7	7	0111	15	-1	1111

負の数の表現（２の補数）

4ビットの場合

+5 (2進表現　0 1 0 1)から -5 の2進表現を得る

0 1 0 1 のビット反転
1 0 1 0 を得る。
それに 1 を加える

```
  1 0 1 0
+)0 0 0 1
  1 0 1 1   ← -5 の2進表現
```

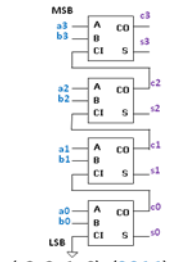
符号無　ud = $b_3 x2x2x2 + b_2 x2x2 + b_1 x2 + b_0$

符号付　sd = $-b_3 x2x2x2 + b_2 x2x2 + b_1 x2 + b_0$

なぜ2の補数表現を用いるのか

2進演算	符号無	符号付
0 0 1 1	3	3
＋ 1 0 1 0	＋ 10	＋ -6
1 1 0 1	＝ 13	＝ -3

2の補数表現をすれば
符号付、符号無で
同じ2進演算アルゴリズム
同じハードウェアを
使用可能



(a3 a2 a1 a0) =(0 0 1 1)
(b3 b2 b1 b0)=(1 0 1 0)

(s3 s2 s1 s0)　=(1 1 0 1)
(c3 c2 c1 c0)　=(0 0 1 0)

負の数の表現(2の補数)

4,5ビットの場合

ビット数の拡張

4 ビット　　5 ビット　　異なる !!

符号無 u0	符号付 sd	2進 b3b2b1b0	符号無 u0	b4b3b2b1b0	符号付 sd	b4b3b2b1b0
0	0	0000	8	01000	-8	11000
1	1	0001	9	01001	-7	11001
2	2	0010	10	01010	-6	11010
3	3	0011	11	01011	-5	11011
4	4	0100	12	01100	-4	11100
5	5	0101	13	01101	-3	11101
6	6	0110	14	01110	-2	11110
7	7	0111	15	01111	-1	11111

2013年10月30日

仕事で成果をあげるために
有効な時間の使い方
強い組織を考える

「マネージメントは実践である」（ドラッカー）

群馬大学大学院　電気電子工学専攻
小林春夫

朝に集中力をもって仕事をする

- 「残業」より「前業」

  朝には　まとまった時間を確保しやすい。

「朝の気は鋭、昼の気は惰、暮れの気は帰 」

(孫子）

- 若者は朝に弱い。

  研究室学生が午前中ゆっくりでも

  あまり自分の考えを押し付けない。

時間の使い方の重要性の金言

時間は誰にでも公平にある。蓄えることはできない。

- 少年老い易すく学成り難し。
  一寸の光陰軽んずべからず。
- 時は金なり
- 先んずれば人を制す

「仕事・回答が早い」ということは価値がある

時間の選択と集中

ドラッカーに学ぶ

- 時間は最も重要な資源である。
- 成果をあげる者は仕事からスタートしない。
  時間からスタートする。
- 自分にとって有効に時間を使えないことには
  「ノー」と言う確固たる決意を持つ。
- まとまった時間で集中的に仕事をする。
  細切れの時間を集めても成果は上がらない。
  （映画を細切れに見ても感動は得られない）
- 他の人にできる仕事は　その人に任せる。
- 仕事に優先順位、劣後順位をつける。

時間管理、集中力

- 忙しい人にこそ急ぎの仕事を依頼する。

忙しい人は時間管理をしっかりやっているので
タイムリーに仕事をやってくれる。

- １０件の仕事を１０日後までに
必ず行わなければならないとき。
簡単で時間がかからない仕事から片付けていく。
（早く仕事の件数を減らしていく）

最後に最も困難な時間のかかる仕事をする。
残っている仕事の件数が少ないとより集中力が高まる。

深く考えることができる環境

- 余裕をもった時間をもつ。

「忙中閑有り」

- 外からの情報をあえて遮断する。
- 日常は「動」の状態。

「静」の状態が「考える」ために必要。

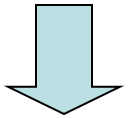
「時に一切の雑念・外界との交わりを遮断し、寺にて座禅を組み瞑想する。」（有識者）

日本の伝統に学ぶ

- 柔道:　相手の力を利用して技をかける
- 剣道:　「残心」 打ち込んだ後も隙がない。
  次にさらに打ち込むという気迫。
- 将棋:

対局開始時には　自分も相手も同型(同じ条件）

過去の対局情報（棋譜）はすべて公開



この状態で強い棋士は常に高い勝率

強い組織にするためには
兵法に学ぶ

- 強みを生かす （ドラッカー）
- 上下　欲を同じくする者は勝つ（孫子）
- 集中が重要。

  局所的に兵力を増す。

  相手を分断し「各個撃破」。

  「戦力の逐次投入」のやり方は敗北パターン。

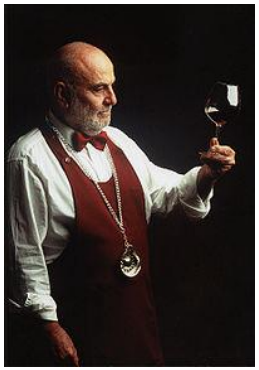
  やるならば最初からどっと戦力を投入する。
- サッカーでのフォーメーション

組織のモチベーションを高める

レンガを積んでいる人に
　「何をしているのか」を問う。
最初の人「レンガを積んでいる」
２番目の人「壁を作っている」
３番目の人「寺院を作っている」

マネージメント：　３番目の人のように、構成員に
　仕事の究極の目的を周知させる。

自分の仕事を理解する



「ソムリエの仕事は
　ワインを説明することではない。
　ワインでお客様を幸せにすることだ。」

ミッション（Mission)

使命、任務

もともとはキリスト教の

伝道、布教、宣教の意

Mission Santa Barbara

El Camino（神の道)

「自主」「独立」の重要性、「個」の確立

- 組織に よりかからない個人のほうが仕事ができる。

  ⇒ 組織にとってありがたい存在

  肩書き、組織名ではなく
  どのような仕事をしているかを
  答えられるようにする。（ドラッカー）

福沢諭吉

- 国を支えて国に頼らず（福沢諭吉）
- 国が諸君に何をしてくれるかではなく、
  諸君が国に何ができるかを問え。(J. F. Kennedy)

餌付けされた動物は野生では生きていけない。

J. F. Kennedy 最後のホテルに偶然にも滞在

2011年3月　米国テキサス州フォートワースで行われた電源回路分野の国際会議 Applied Power Electronics Conference and Exposition (APEC2011) に参加。

滞在先のヒルトン ホテル　(当時　ホテル テキサス) が J. F. Kennedy（当時）大統領の最後の宿泊先であることを、会議に一緒に参加した恩田謙一客員教授より告げられる。

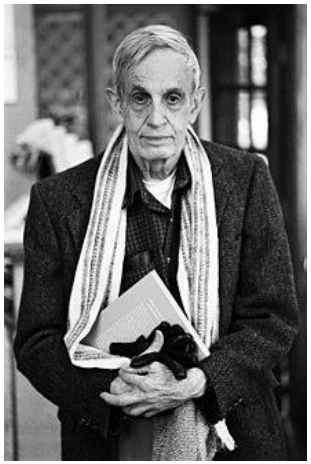
ホテルのロビーには　J. F. Kennedy の像と写真が展示。1963年11月21日ホテルテキサスに宿泊し、同ホテルで翌朝最後となった演説を行い ダラスに向かったとある。

個人でも組織でも「分野でトップになる」

強いものはますます強くなる
Winner Takes All

- ランチェスターの法則（軍事）
- ゲーム理論

ジョン・ナッシュの理論 ⇨ 「独占禁止法」の考え方へ



John Nash
1928-2015 米