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).

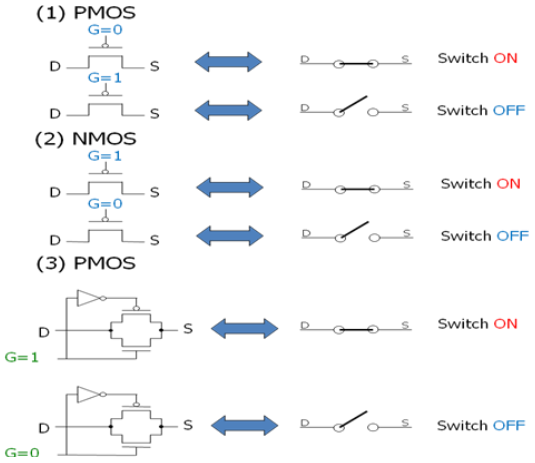


図 1　PMOS, NMOS, CMOS スイッチ

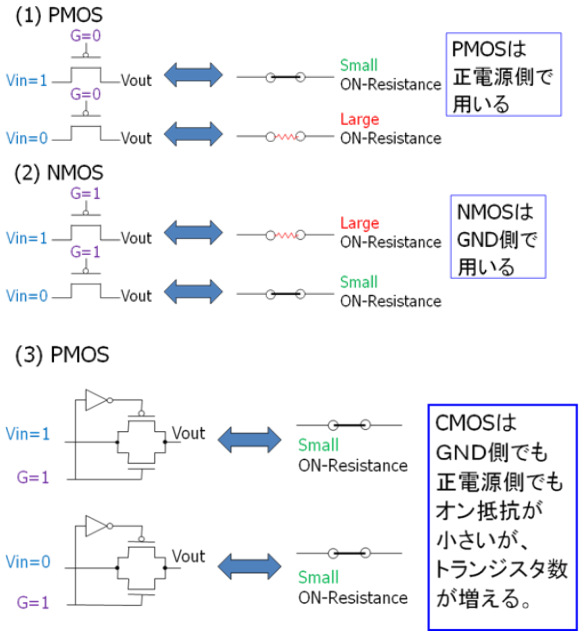


図 2　PMOS, NMOS, CMOS スイッチのオン抵抗

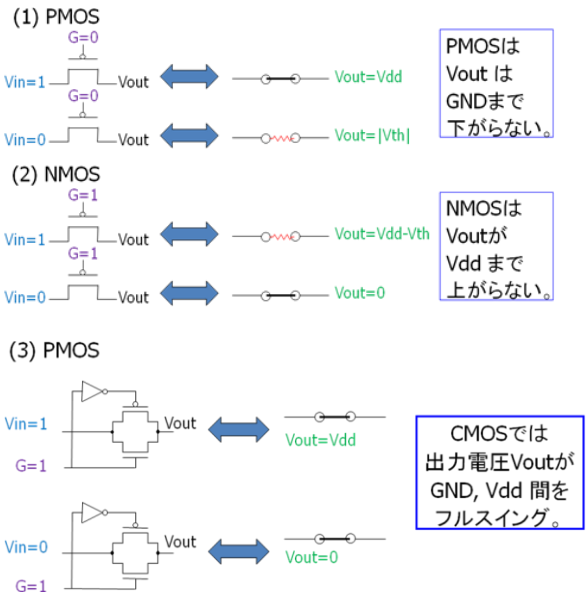


図 3 PMOS,NMOS,CMOS スイッチの出力電圧

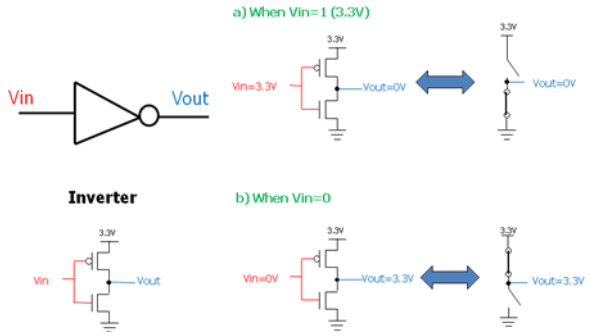


図 4　CMOS Inverter 回路

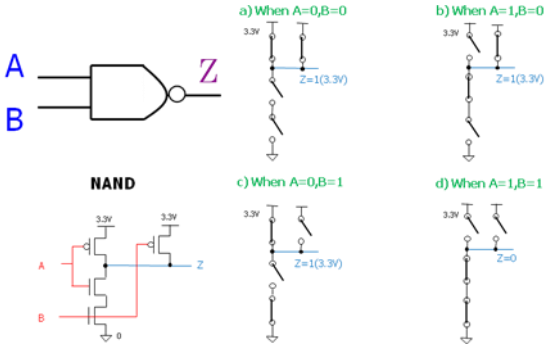


図 5　CMOS NAND 回路

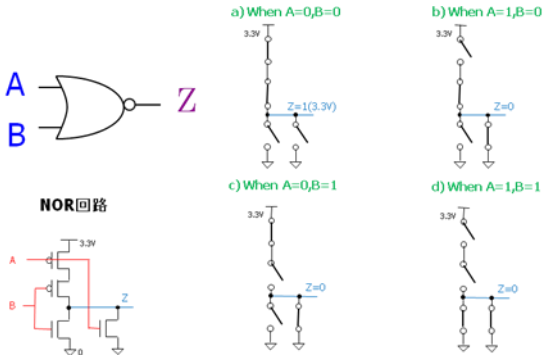


図 6　CMOS NOR 回路

	論理積	論理和
PMOS	並列	直列
NMOS	直列	並列

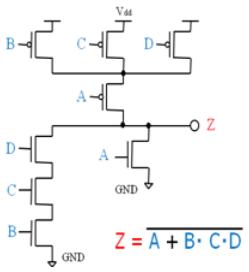


図 7　CMOS 複合ゲートの回路構成の規則と回路例

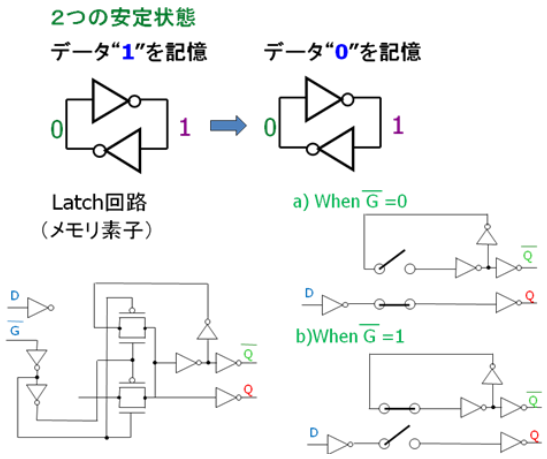


図 8　CMOS ラッチ回路

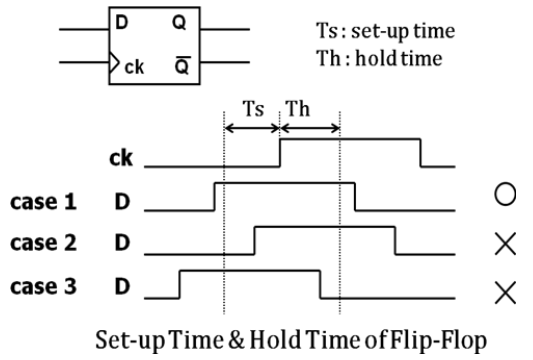


図 9　Flip-Flop 回路インターフェース仕様

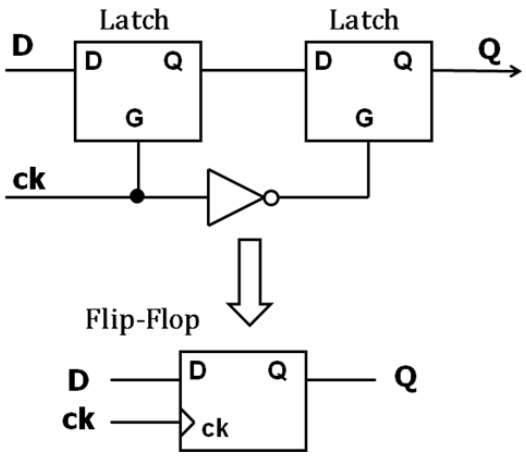


図 10　Flip-Flop 回路の構成

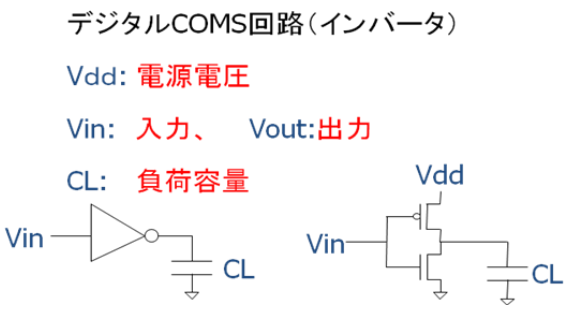


図 11 負荷容量 CL

（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

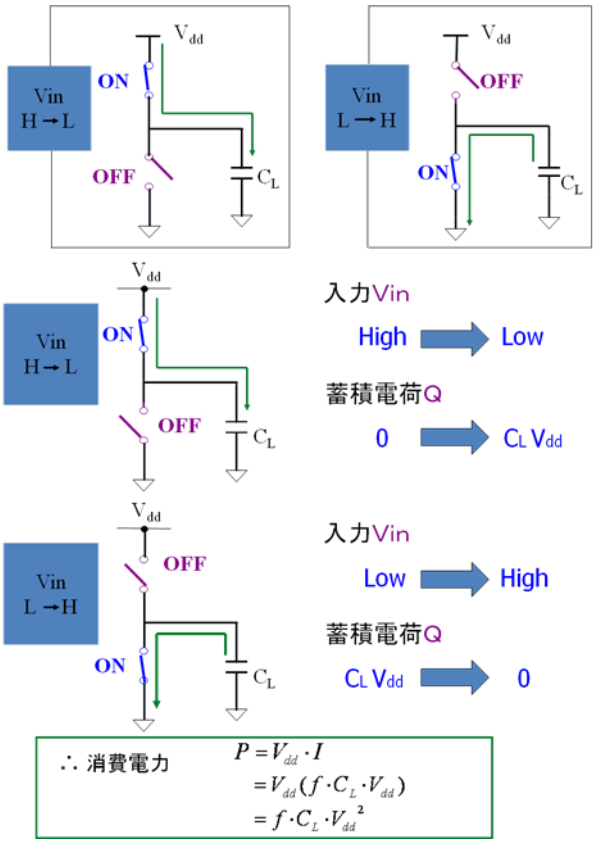


図 13　CMOS インバータ動的消費電力

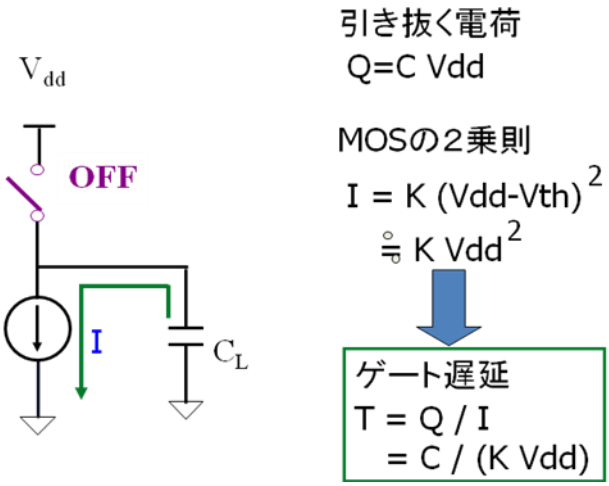


図 14　CMOS インバータの遅延（スピード）

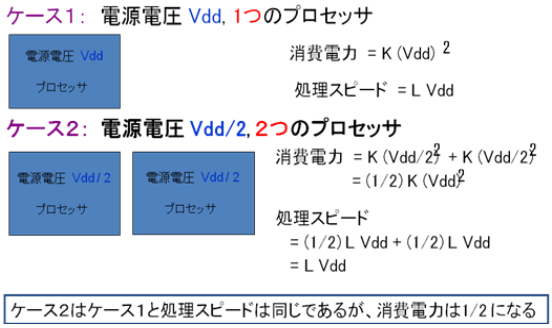


図 15　マルチプロセッサ構成による低消費電力化

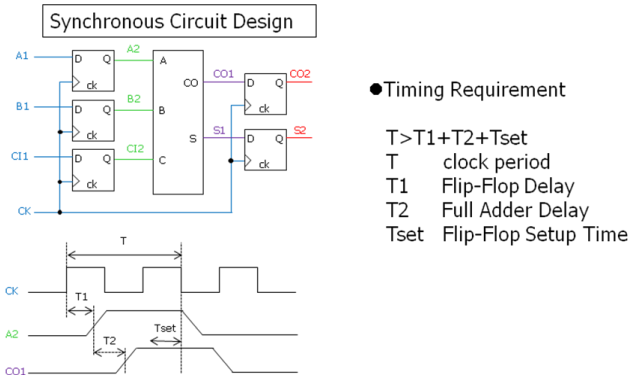


図 16　同期回路とタイミング設計

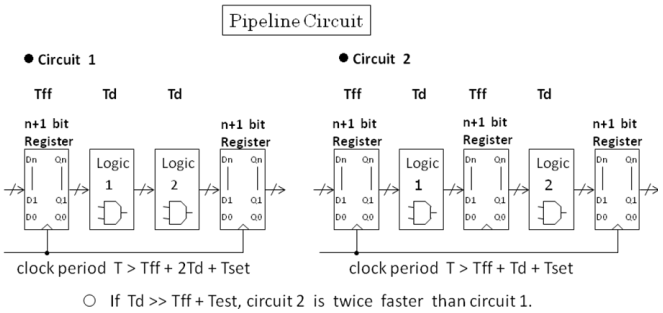


図 17　パイプライン回路

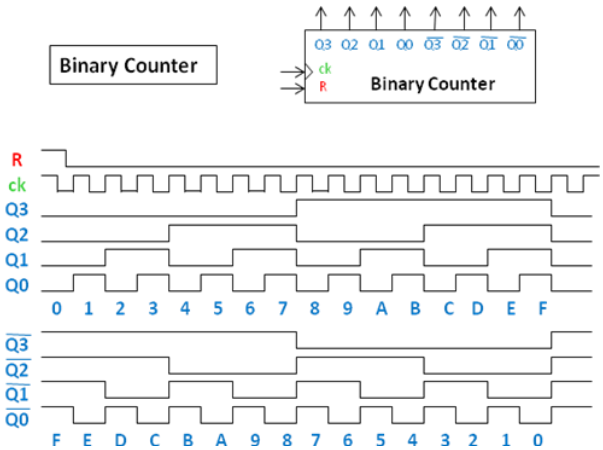


図 18　2 進カウンタとタイミングチャート

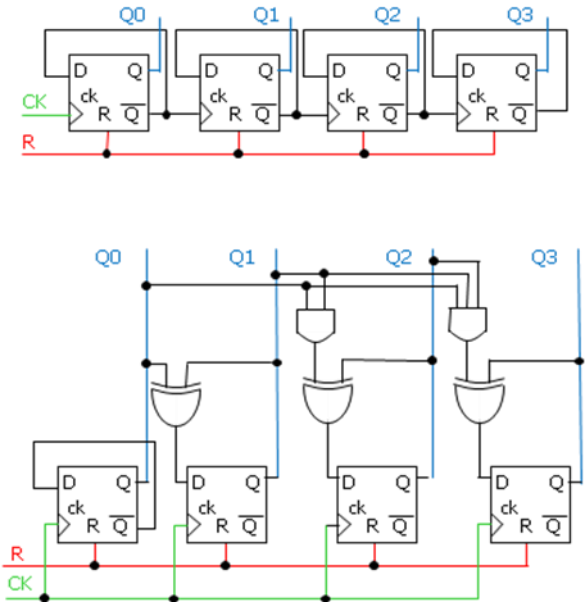


図 19　（上）非同期、（下）同期カウンタ回路

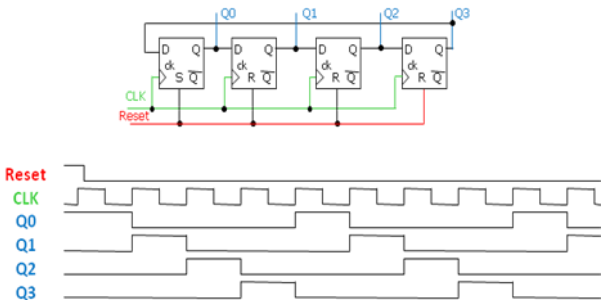


図 20　リングカウンタ

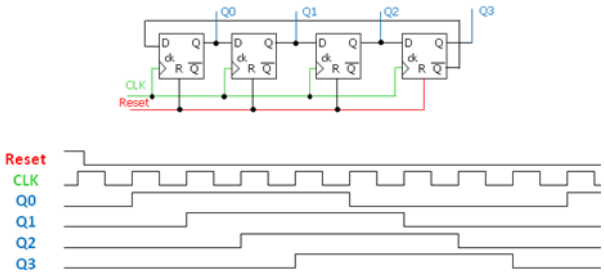


図 21　ジョンソンカウンタ

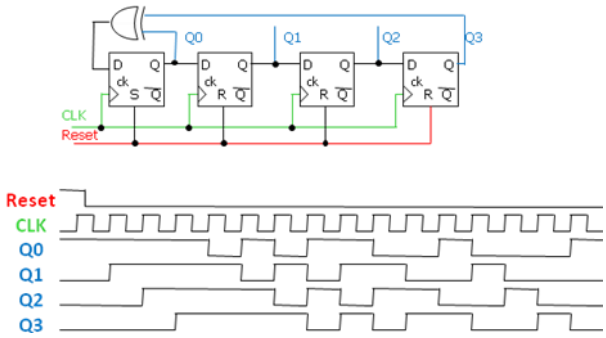


図 22　M 系列発生回路（LFSR）

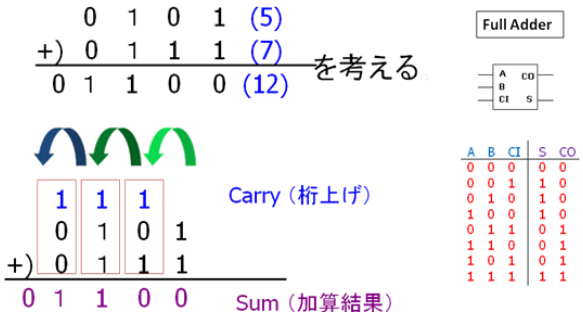


図 23　２進数加算アルゴリズムと Full Adder

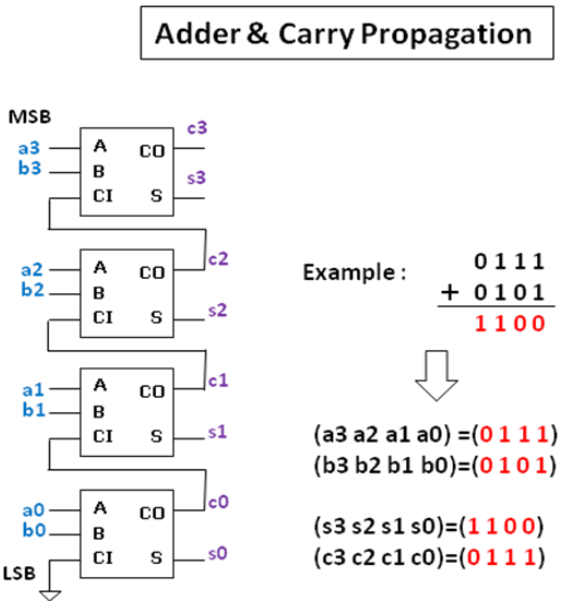


図 24　4bit Carry Ripple Adder

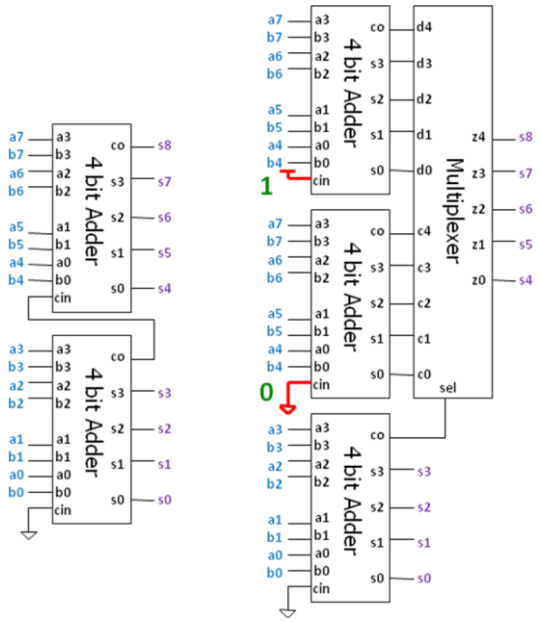


図 25　8bit Adder　（左）通常構成

（右）8bit Carry Select Adder

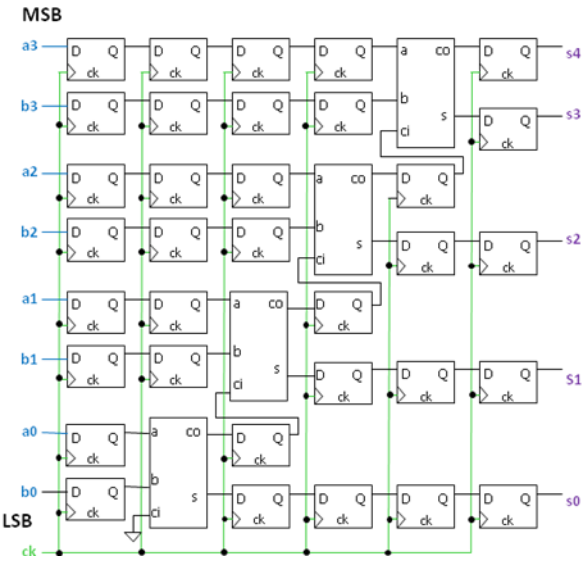


図 26 (a)　パイプライン加算器の構成

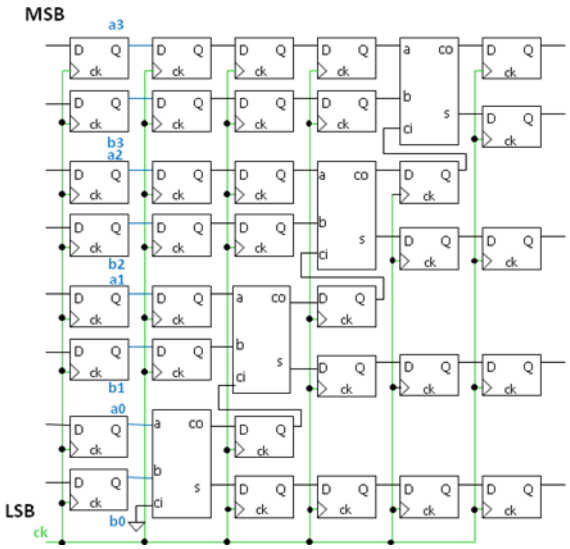


図 26 (b)　パイプライン加算器　１クロック目

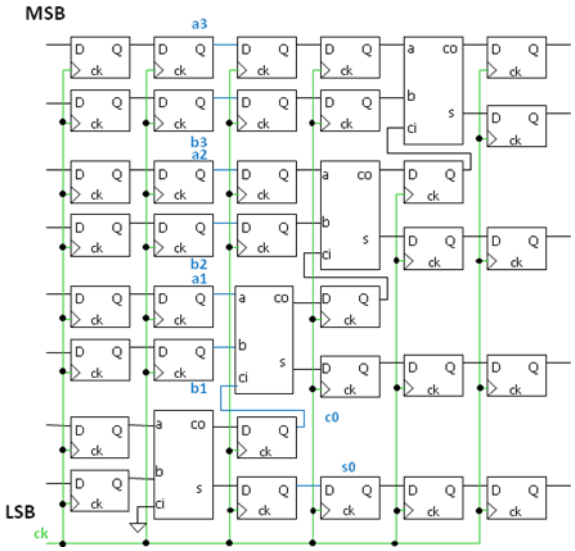


図 26 (c)　パイプライン加算器　２クロック目

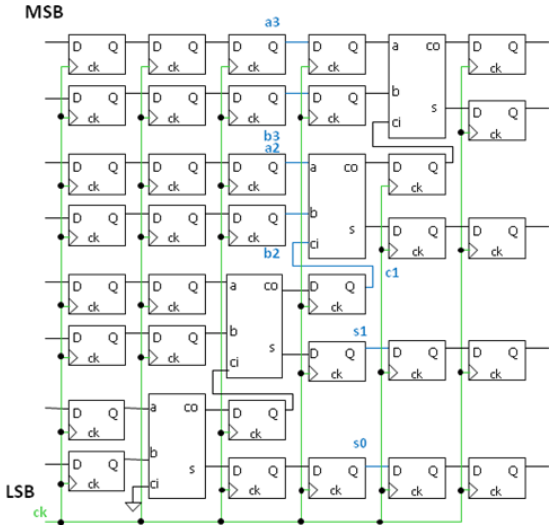


図 26 (d) パイプライン加算器　3 クロック目

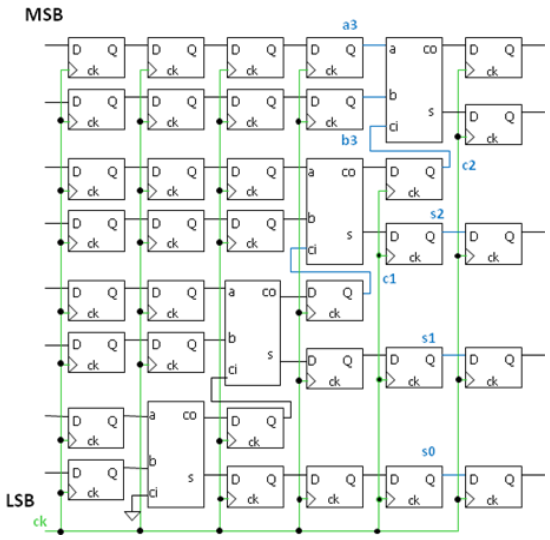


図 26 (e) パイプライン加算器　4 クロック目

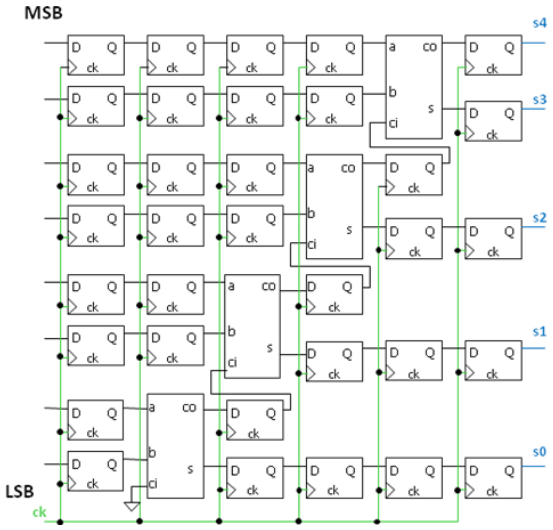


図 26 (f) パイプライン加算器　5 クロック目

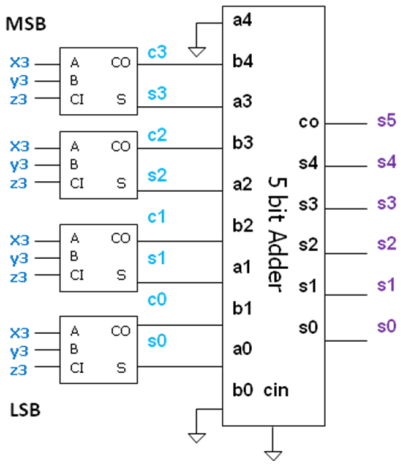


図 27　Carry Save Adder (S=X+Y+Z)

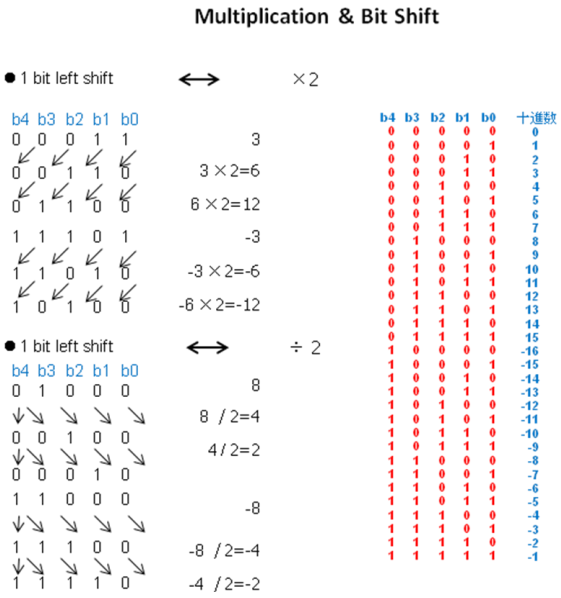


図 28　ビットシフト

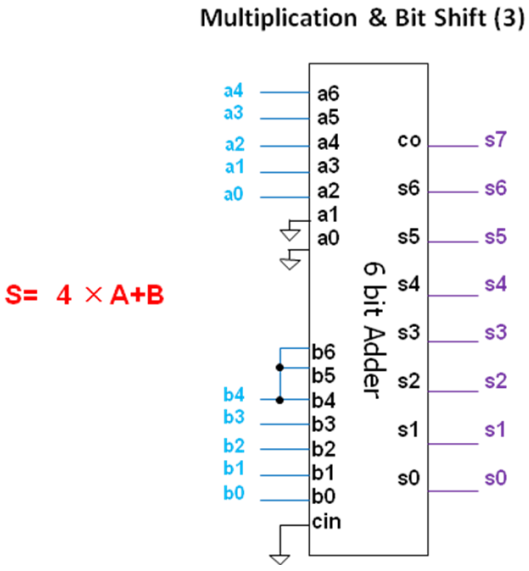


図 29　ビットシフトと加算

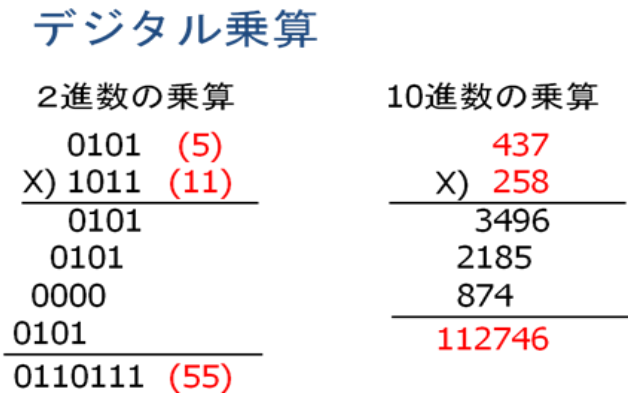


図 30　乗算アルゴリズム

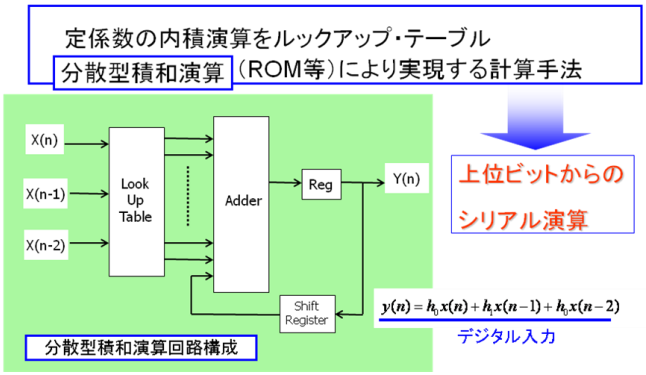


図 31　分散型積和演算器構成

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

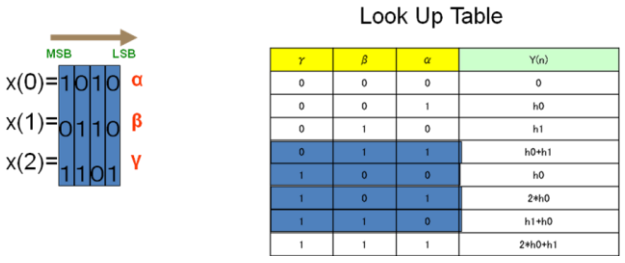


図 32　ROM 内部の説明

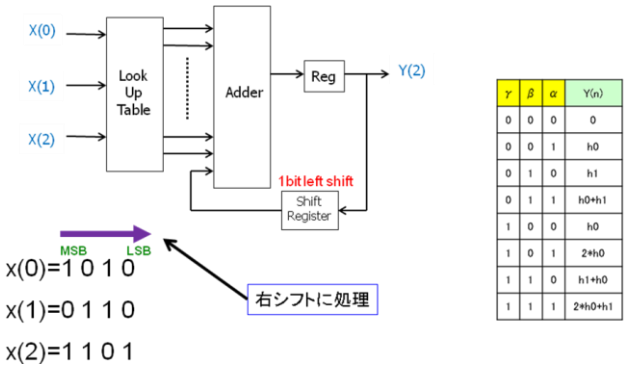


図 33 (a)　分散型積和演算器動作

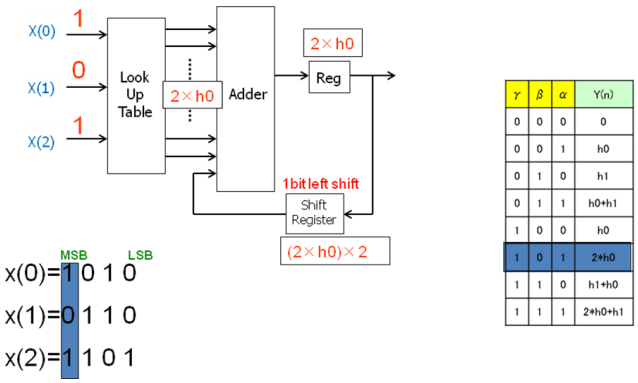


図 33 (b)　分散型積和演算器動作 1

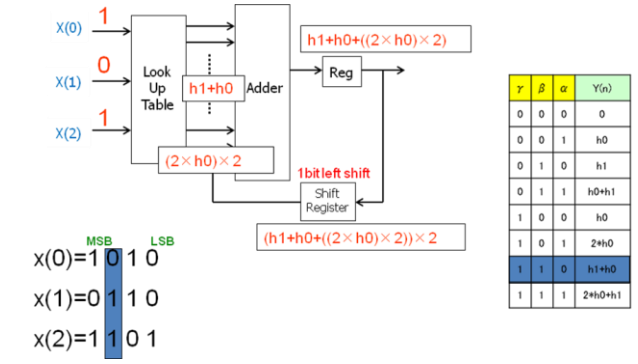


図 33 (c)　分散型積和演算器動作 2

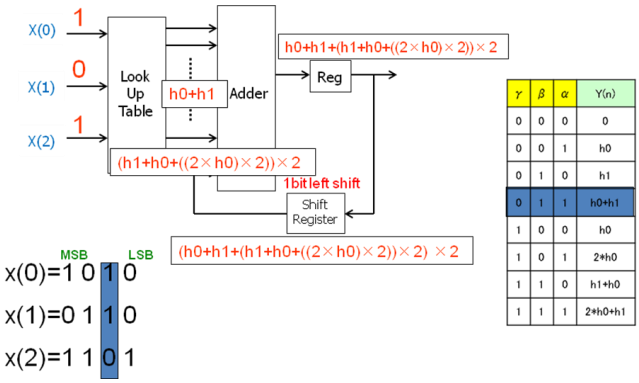


図 33 (d)　分散型積和演算器動作 3

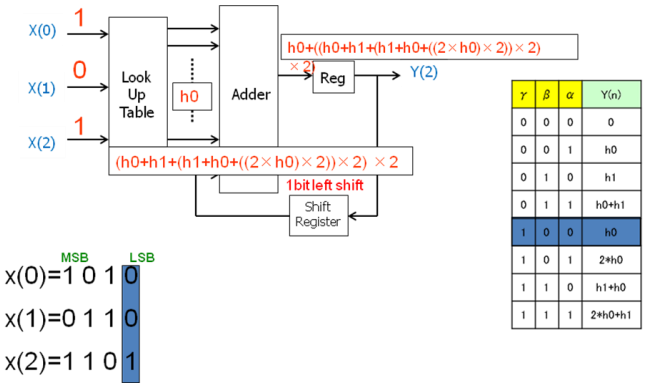


図 33 (e)　分散型積和演算器動作 4

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
```

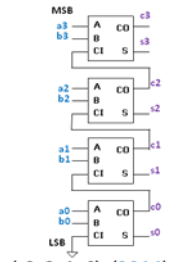
符号無　ud = $b_3 x2x2x2 + b_2 x2x2 + b_1 x2 + b_0$

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