はじめに

計算機の歴史は非常に古く、ハード面では中近東のバッカスといういわゆるソロバンが始まりだと言われています。東方に伝わってソロバンと呼ばれるようになりましたが、西に向かった方は後に機械式の計算機から真空管式、次の半導体式の計算機、最初はアナログ計算機後に次にディジタル計算機へと変化しました。

ソフトウエア面においては、ソクラテスによる伝統式形式論理学、次に記号論理学が発展し、その中でブール代数など生まれました。その中でガウスやペアノなどの数学者によるなどによる整数論なども取り入れられ発展してきています。

用途は砲弾の弾道計算から始まり、IBM による事務計算などが高速で行えるようになり、１９００年ごとから急速に発展し、現代では AI など用途は多方面に渡っています。Ai に関して１９６０年頃は人工知能と呼ばれ自ら考えるという意味でしたが、近年では「考える能力」ではなく「多数の事例から最適な解答を見つけ出す」ということを AI と呼んでいるようです。AI の定義は難しいと思えますが、本来の意味は未だに謎なのかもしれません。

この著書はディジタルを始める人の最初の取っ掛かりとして、基本的な事柄を中心にまとめたつもりです。ディジタルの世界はものすごい勢いで今も発展しており、様々な分野に分かれてしかもその各分野において多数の新事実が見いだされているため、全ての知識は膨大なものになります。しかしディジタルの基本はすべての分野に渡って共通していますので、いずれの

ディジタルの分野に携わっていく人にも役に立つように構成したつもりです。貢献できることを願っております。

２０２３年　　新原　盛太郎

目次

図目次

表目次

第Ⅰ部

基礎編

第 1 章

はじめに

ディジタルの勉強を始めるに当たって、 アナログの世界がありながら なぜディジタルを用いなければならないかと言うことを知ることは大切です。人間が扱う現象がアナログですので、何もわざわざディジタル信号へ変換して扱う必要がないような気もします。しかしディジタルにすることによって、今まで得られなかったような新しい世界が現れてくるのです。その新しい世界とは、どの様なモノであるか、ここでは述べています。

1.1　ディジタル回路の歴史

ディジタル回路の学問としての歴史は、それ程古い話ではありません。勿論ディジタルと意識しないで、ディジタルを扱っていた歴史自体は古いものがあります。例えば太鼓をたたいて信号をやり取りをする、あるいはのろしを上げて連絡し合うなどと言ったことは、ディジタルという意識は全く無かったと思われますが、信号の種類から言って完全にディジタル信号です。この様に無意識ながらも人類とディジタルの繋がりは古い歴史を持っていますが、ディジタルということをはっきりと意識し始めたのは、いわゆる論理代数の中のブール代数が始まったころからだと思われます。ディジタルは、このブール代数を基礎として構成されています。

ブール代数は、数学の一分野であり、その起源はソクラテス、プラトン、アリストテレスの時代までさかのぼることが出来ます。当初は伝統的形式論理学と現在呼ばれている、いわゆる論理学で、その一例を挙げますと

[犬] は [動物] である。真か偽か。

というような文章において、括弧の中身はどの様なものであっても良いのです。これは文章の形式のみに注目する、いわゆる形式の真理を追究する学問です。論理学に興味がある方は、次の文献などをご覧ください。[6]

その次に現れたのは、記号論理学と呼ばれる学問で、形式論理学において出てくる命題に対して記号を適用して研究していった学問です。この流れの中からいわゆるブール代数が生まれてきました。ブール代数は 0 と 1 しか無い。つまり真か偽かと言うことです。よって過去の論理学の成果の大部分を利用することが可能です。論理学そのものは、そちらの方へ譲って、この著作ではブール代数から始めることにします。

その他のディジタルに関連した学問としては、集合論や整数論があります。集合論は、ブール代数に関連したところだけ取り扱うことにします。整数論に関してはただ単に 0 と 1 のみでは無く、もっと多数のとびとびの値を用いる学問ですので、この著作で学ぶディジタルにおいては、とりあえず必要がありません。整数論に興味がある方は、次の文献などをご覧ください。[7]

1.2　ディジタルの利点

全ての信号処理はアナログ回路を用いて実現することが可能です。ではなぜディジタル回路がこの様に発展してきたのでしょうか。それには次の事柄があげられます。プログラムすることが可能です。安定性がアナログに較べて優れている、再現性がある、適応アルゴリズムを用いることが出来る、誤り訂正機能を持たせることが出来る、特殊な線形位相フィルタを実現することが出来るなどなどです。ここではこの言葉だけに止めておきますが、詳細

については、文献 [1] を参照願います。

1.3 ディジタル集積回路

ディジタルＩＣの分類方法として TI(Texas Instrument) 社において考案され、今や世間一般で使われている分類方法を述べておきます。

LSI(Large-Scale Integration　100 個以上のゲートあるいは同等の複雑さを持つＩＣ

MSI(Medium-Scale Integration)　12 個以上のゲートあるいは同等の複雑さを持つＩＣ

SSI(Small-Scale Integration)　MSI よりも少ないゲートあるいは同等の複雑さを持つＩＣ

VLSI(Very-Large-Scale Integration)　1000 個以上のゲートあるいは同等の複雑さを持つＩＣ

1.4 ディジタル回路の概要

ディジタル回路は、ブール代数を回路に適用して実現されています。多くのディジタル回路設計者は、回路そのものつまりトランジスタや抵抗、容量、コイルといった素子そのものを取り扱うことは、ほとんどありません。ブール代数を回路として実現するための最小単位のセル（例えてみれば箱）を並べていくことによってディジタル回路を実現していく場合が多いようです。つまりディジタル回路設計者は、電気電子回路の知識を必要としなくても実現することが出来ます。むしろ知らなければならないことは、世の中の現象をいかに代数の形に置き換えるか、次にその代数をどうやって箱を使って実現していくかと言うことです。この作業自体も膨大な労力と時間が必要になります。よっていかに効率良く作業量を減らし、しかも質を下げずに時間を短縮するかが重要になります。その目標に一歩でも近づけようとするの

がこの著書の目的です。

1.5　ディジタルの長短

ディジタル化することによって、何のメリットもないとするならば、ディジタル化は単なるお話で終わってしまうことでしょう。しかしディジタル化というのは、アナログの世界では得られない様なすばらしいメリットを持っています。これらのディジタルが持っているすばらしいメリットあるいはデメリットについて、これから考えていくことにします。

安定である

通常ディジタル回路は、温度に対して十分に余裕をもって設計してありますので、多少の温度変化やその他の変動に対して全く影響を受けません。その結果経時変化に対してもアナログに比較して非常に安定な回路が得られます。これはディジタル回路の信号が 0 と 1 の二つの信号しか必要でないと言うことで、あるスレッショルド・レベルを超えない限り 0 と 1 の判定を誤ることがないためです。このことはディジタルであることの大きなメリットです。

再現性がある

同じ現象に対してディジタル回路では、何度でも全く同じ結果が得られます。しかしアナログ回路の場合には、回路に使っている部品のバラツキによって結果が微妙に異なってきます。

またディジタルの場合には、情報が欠落してもその欠落の程度が小さければ完全の元の状態に戻すことが可能です。具体的な例としてレコード盤と CD とを考えてみるとよく分かります。 レコード盤の場合には、ゴミがついていたり小さな傷があるとその影響で雑音が発生します。これに対して CD の場合には、多少のゴミや傷があっても元の信号を再生する能力を簡単に持たせることが出来るため、雑音は全く生じません。

実現が容易

特に低周波において、アナログ回路に較べてディジタル回路は実現することが非常に簡単です。ディジタル回路の場合には、何処かと何処かが接続されていれば確実に動きます。よってアナログの様に配線の引き回しはほとんど考える必要がありません。

しかしディジタル回路は、パルス信号を 扱いますので、パルスの立ち上がりと立ち下がりの所には高い周波数成分を含んでいます。この高い周波数成分が、別の回路に対して悪い影響を与えることがあります。

セル化が容易

ディジタル回路は、アナログ回路に比較してセル化が容易です。これは扱う信号が 0 と 1 しかないことから生じています。ディジタル回路の発展は、このセル化が容易であるという特長を生かして大規模化が進んできました。最初はインバータや AND、OR、NOR、NAND 等の基本素子から始まり、メモリ、MODEM などのやや大きなセル、 システム規模のセルと言うような発展です。

この様に回路を分割することが容易であるため、ある特殊な部分をソケットのように差し替えて使うことも可能となります。

データの蓄積が容易

ディジタルは繰り返し述べていますように 0 と 1 の集まりに過ぎませんので、これらを保存することも容易です。たった二つの状態がありさえすればよいので、あらゆる形で情報を保存することが可能となります。二つの状態は、この自然界ではあらゆる所に存在するからです。

さらに蓄積されるデータは、0 と 1 だけですからこれらの情報が失われてしまう危険性も連続関数に較べて少なくなることは、容易に想像できると思います。

データの圧縮が容易

全てのデータは、情報自体が持っている特殊な性質[*1]を利用して信号を圧縮することが出来ます。

この圧縮をするときに二つの方法が採られています。一つは圧縮した信号が完全に元に戻すことが出来る圧縮方法、もう一つは完全には戻らない圧縮方法です。前者の方法として、ZIP ファイル、 後者の方法として JPG などがあります。 これ以外にも非常に多数の方法が存在していますし、現在も研究が続けられています。

処理速度の限界

ディジタルが注目されだしたのは、1980 年初頭に過ぎず、歴史として非常に新しい技術です。この理由は、ディジタル信号がパルスを扱って表現しているため、高い周波数成分まで増幅する必要があるためです。つまり素子の性能が追いついていなかったことがディジタル信号処理の発展を押さえていたと言うことです。音声の場合には、扱わなくてはならない周波数が音声信号として 20 $[kHz]$ しかありませんので、ナイキストの定理から 40 $[kHz]$ の周波数でサンプリング すれば良いことになります。例えば CD の場合のサンプリング周波数は、44.1 $[kHz]$ です。これはあくまでもサンプリング周波数です。このサンプリングされた周波数を量子化によってパルス に変換しなければなりませんので、パルスを表現するためにこの数十倍の周波数帯域が必要となります。この様に考えてきますと素子の周波数限界を向上することによってのみディジタル化が実現できると言うことが分かります。その結果ディジタル・テレビの実現は、2000 年の始めになってようやく可能となりました。

この様にディジタル化の歴史は、素子の性能限界によって大きく左右されております。現代のディジタル・テレビであってもディジタル化されている

[*1] 例えば音声の場合、極端に周波数が高い音や低い音は聞こえないなど。

のは、ベースバンド領域のみです。[*2]より高い周波数がディジタル化できるようになってきますと、さらに多様なディジタル化の世界が広がってくると思われます。

1.5.1 アナログの世界

アナログ技術は、古くから現在に至るまで研究が続けられ、今も新しい話題を提供し続けています。ディジタルにかなりの部分を奪われてはいますが、アナログの世界が終わったわけではありません。日本においては、1980 年頃からアナログ技術者がディジタルへと転向し、大学などの研究機関もディジタルの世界へシフトしてしまいましたが、欧米においてはアナログの勢いはそのまま継続して進行しています。その結果日本のアナログ技術は世界に対して大きく遅れてしまったと言わざるを得ません。例えば Translinear、電流モード設計、CCII などの基本技術は、日本において全くと言って良いほど研究がなされていません。

アナログ回路の発展に関しては、日本は大きく遅れてしまっています。しかしこの遅れは、必ず取り戻さないとあらゆる面で将来大きな弊害が出る可能性があります。またディジタル回路で何らかのトラブルが発生したときに、対応することが出来るのはアナログの知識が必要です。そういった意味からもアナログの研究は必須の研究対象です。

[*2] ベースバンドとは、人が検知できる周波数帯のことを指します。

第 2 章

数の体系

ディジタル回路の具体的な話にはいる前に、数についての話を始めます。一般的に多くの人は１０進数に親しんでいるので、１０進数とは違った数体系についてしり込みをしてしまいます。しかし１０進数以外に日常使っていないかというとそうでもありません。例えば時刻の場合は６０進数であり、日にちの数え方は３０進数あるいは３１進数を用いています。

ディジタル回路においてよく用いられる数体系は、２進数、８進数、１６進数です。これらはその便利な性質によって選ばれています。つまりディジタル回路において１０進数というのは、実現する場合において取り扱いが難しいということから上のような数体系が用いられているのです。

2.1　ディジタル回路の数体系

２進数は 0 から始まり、1 で終わる二つの数字から構成され、更に 1 増えると 10 となる数体系であり、同様に８進数は 0 から 7 まで、１６進数は 0 から 9 までの数字と１０進数の 10 から 15 までは、A、B、C、D、E、F からなる数体系です。１０進数、８進数、１６進数についての対応表を表 2.1 に示しておきます。

１０進数	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
８進数	0	1	2	3	4	5	6	7	10	11	12	13	14	15	16	17
１６進数	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

表 2.1 対応表

２進数、８進数、１６進数の和、差、積、商は１０進数と同じですが、１０進数とは違った印象を与えるため、その計算においては、注意が必要です。ここでは２進数を中心に説明をしていくことにします。このような n 進数と呼ばれる数体系を基数（radix、あるいは底：base）n に基づく数体系と呼んでいます。

一般的に n 進数は、0 から $n-1$ 個の数字が必要になります。１０進数と n 進数との関係は、次のように与えられます。

$$N = a_0 n^0 + a_1 n^1 + a_2 n^2 + \cdots + a_m n^m \qquad (N \geq 1) \qquad (2.1)$$

$$M = a_{-1} n^{-1} + a_{-2} n^{-2} + \cdots + a_{-m} n^{-m} \qquad (0 < M < 1) \qquad (2.2)$$

ただし

$$m = 0\ ,\ 1\ ,\ 2\ ,\ \cdots \infty$$

$$0 \leq a_k \leq n-1$$

2.2 加減乗除

２進数、８進数、１６進数の加減乗除について説明を行います。加減乗除にのみ限定して話をすれば、基本的に１０進数と何ら変わるところはありません。

２進数

これまで説明してきましたように、２進数は 0 と 1 だけで出来上がっている数体系です。たった二つしか数字が存在しないので簡単であるような印象を与えますが、実際は非常に複雑な体系であることが次第に分かってき

ます。

２進数の和、差、積、商を、どの様な方法によって行えば良いかと言うことについて述べていきます。

まず始めに和について考えると２進数では 2 という数字は存在しないので、1 の次は 10 となることに注意すれば結構です。例を挙げると、次のようになります。

$$\begin{array}{rr} & 111 \\ + & 11 \\ \hline & 1010 \end{array}$$

和の場合について述べておきますと、差の場合はその逆を行うだけですので、とり分けて説明しなくても良いでしょう。

積の場合は、1 と 0 を掛けると 0 であり、1 と 1 とを掛けると 1 になることと、その後は和の計算になるので、この場合もさして困難なく計算できると思います。

$$\begin{array}{rr} & 111 \\ \times & 11 \\ \hline & 10101 \end{array}$$

商の場合は、和、差、積の知識が必要になりますので少々複雑ですが、この場合も普通の１０進数での計算方法と変わるところはありません。積の場合に用いた式を逆に商の場合に適用してみると分かると思います。

８ or １６進数

８進数、１６進数の加減乗除も全く２進数と変わるところがありません。気を付けなければならないことは、８進数の場合には、9 になると桁上がりになること、数が不足した場合には上の桁から数字を下ろしてくるとき、9 であることに注意しさえすれば結構です。

１６進数の場合には、0 から F までの数字であることに注意しさえすれば、その他の数体系と全く同様にして考えることが出来ます。

2.3　相互変換

これまで出てきた１０進数、２進数、８進数、１６進数の間の相互変換について考えます。特に１０進数と２進数の相互変換、２進数と８進数の相互変換について考えることにします。

まず始めに１０進数と２進数との相互変換ですが、これらの数体系の変換を考えるためには、(2.1) 式と (2.2) 式とが重要です。これらの式の左辺は数字一つですので、ｎ進数から１０進数への変換は、右辺の式へ代入して計算すれば直ちに求められます。

問題は逆の場合です。つまり１０進数からｎ進数への変換をどの様にして行えばよいかと言うことです。式が二つ (2.1) 式と (2.2) 式がありますので、ここでも二つの場合について考えていきます。

$N \geq 1$ の場合

この場合 (2.1) 式から係数 a_k は、次のように求められます。

$$
\begin{aligned}
a_0 &: \frac{N - a_0}{n} = a_1 + a_2 n + a_3^2 + \cdots + a_m n^{m-1} = N_1 \\
a_1 &: \frac{N_1 - a_1}{n} = a_2 + a_3 n + a_4 n^2 + \cdots + a_m n^{m-2} = N_2 \\
a_2 &: \frac{N_2 - a_2}{n} = a_3 + a_4 n + a_5 n^2 + \cdots + a_m n^{m-3} = N_3 \\
a_3 &: \frac{N_3 - a_3}{n} = a_4 + a_5 n + a_6 n^2 + \cdots + a_m n^{m-4} = N_4 \\
&\vdots \\
a_{m-1} &: \frac{N_{m-2} - a_{m-2}}{n} = a_{m-1} + a_m n = N_{m-1} \\
a_m &: \frac{N_{m-1} - a_{m-1}}{n} = a_m
\end{aligned}
$$

この式を続けていけば係数が順に求まりますが、この式の中で左辺の N_{-k} の後の引き算で出てくる数値が係数となります。２進数の場合 a_k の値は 0

でなければ全て 1 となります。計算によって低い冪の係数から計算されることになりますが、最後に残った計算結果が 0 の場合は 0 となりますが a_m が残った場合は、一番大きな係数の値が a_m となることを忘れてはなりません。

このように次々と割っていき、その余りを求めていくことによって n 進数の係数を求めることが出来ます。

例として 33 の 2 進数表示を求めてみます。次の計算は 2 で順に割っていくのですが、一番右に割り算の残りを示す形で計算を進めています。このほうが先の説明の式より実際に計算を行う場合に分かりやすいと思います。この結果は、次のようになります。

$$\begin{aligned}\frac{33}{2} &= 16\cdots 1\\ \frac{16}{2} &= 8\cdots 0\\ \frac{8}{2} &= 4\cdots 0\\ \frac{4}{2} &= 2\cdots 0\\ \frac{2}{2} &= 1\cdots 0\end{aligned}$$

以上の計算から $33_{(10)} = 100001_{(2)}$ という結果が得られます。添え字の数字は、各々１０進数、２進数であることを示しています。

$0 < N < 1$ の場合

この場合には (2.2) 式を用いて、両辺に n を掛けて式を変形すると次の結果が得られます。

$$
\begin{aligned}
***&: Mn = a_{-1} + a_{-2}n^{-1} + a_{-3}n^{-2} + \cdots + a_{-m}n^{-m+1} = M_{-1} \\
a_{-1}&: M_{-1}n - a_{-1} = a_{-2} + a_{-3}n^{-1} + a_{-4}n^{-2} + \cdots + a_{-m}n^{-m+2} = M_{-2} \\
a_{-2}&: M_{-2}n - a_{-2} = a_{-3} + a_4 n^{-1} + a_5 n^{-2} + \cdots + a_{-m}n^{-m+3} = M_{-3} \\
a_{-3}&: M_{-3}n - a_{-3} = a_{-4} + a_{-5}n + a_{-6}n^{-2} + \cdots + a_{-m}n^{-m+4} = M_{-4} \\
&\vdots \\
a_{-m-1}&: M_{-m+2n} - a_{-m+2} = a_{-m+1} + a_{-m}n^{-1} = M_{-m+1} \\
a_{-m}&: M_{-m+1}n - a_{-m+1} = a_{-m}
\end{aligned}
$$

この式を見て分かることは、式の左辺の第 2 項が求める係数となります。このように n を掛けていって小数点より小さい数字は、その係数を求めることが出来ることが分かります。この式を使って係数を求めることはけっこう大変です。そこで次の例を考えて、係数を求める方法を示していきます。

$0.6875_{(10)}$ を 2 進数に変換する問題を考えてみますと、次のように求められます。計算方法として 2 を掛けた後に小数点以上の数値が出たら、その数値を左側に記述し、残りの数値に更に 2 を掛けて計算を続けることです。

$$
\begin{array}{rcr}
 & & 0.6875 \\
 & & \times 2 \\
\hline
= a_{-1} & 1 & 0.3750 \\
 & & \times 2 \\
\hline
a_{-2} = & 0 & 0.7500 \\
 & & \times 2 \\
\hline
a_{-3} = & 1 & 0.5000 \\
 & & \times 2 \\
\hline
a_{-4} = & 1 & 0.0000
\end{array}
$$

この結果から、$0.6875_{(10)} = 0.1011_{(2)}$ が得られます。

これら二つの変換の係数を求めるときに、小数点の近くから離れる方向に向かって順番に係数が求められることが分かります。

これらの変換は特に何進数への変換と言っていないので、８進数でも１６進数でも同様に適用することが出来ます。

次に２進数と８進数との関係について考えることにします。８進数は０から７までの数字で構成されています。８進数の７は２進数では 111 と表すことが出来ます。つまり２進数の３桁によって総ての８進数の数字が表現できていることが分かります。つまり２進数で３桁ごとに区切って、その区切られた各々について８進数の数字へ変換してやれば、そのまま８進数を表現していることになります。

８進数から２進数への変換も非常に簡単で、８進数で表してあり、数字をその桁ごとに２進数へ変換して並べるだけで直ちに２進数が得られることになります。

１６進数の場合も８進数と同様です。ただ１６進数の場合２進数を４桁まで使えば良いことになります。逆の１６進数から２進数への変換も、１６進数の各桁数に２進数の４桁を当てはめていけば、変換できることになります。

2.4 コード化

情報を計算機で取り扱うためには、その情報を何らかの形で数値化する必要があります。その数値化の手法がコード化と呼ばれているものです。そのコード化の手法は、一つの情報が一つのコードに対応し、お互いに区別さえ出来れば良いわけですので、様々なコード化が考えられます。

以上のことからいかようなコード化も可能であることから、簡単であること、コストが安くなることなどの要求を満足するようにコード化されることが望ましいことになります。コード化の代表的なものについて述べていくことにします。

２進コード

２進コードとは、単に１０進数から２進数へ変換しただけのコードである。その対応表を次に示しておきます。

１０進数	２進数
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
…	…

表 2.2　２進コード

2.4.1　ＢＣＤコード (Binary-Coded Decimal)

１０進数の各桁ごとの数値に、２進数のコードを当てはめるのがＢＣＤコードです。１０進数を２進数で当てはめるためには、４ビット必要になります。

４ビットは１６通り表示できますので、そのうち１０個の２進数を用いることになります。お互いに違った表示であればどの２進数を当てはめても良いのですがどの２進数を当てはめるかと言うことについて、様々なＢＣＤコードが存在します。

2.4.2　8-4-2-1 ＢＣＤコード

このコードは２進数の最初の１０個を使うコードであり、次のように対応しています。

１０進数	8-4-2-1 BCD
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

表 2.3　8-4-2-1 BCD コード

ここで勘違いをしないように、$369_{(10)}$ の 8-4-2-1 BCD コードを求めておきます。

3	6	9
0011	0110	1001

これから $369_{(10)} = 001101101001_{(BCD)}$ となることが分かります。

逆に BCD コードから１０進数へ変換するには、一番下の桁から４つずつ区切って、その区切られた数値に対して $8-4-2-1$ の重みを付けて１０進数へと変換すればよいことになります。

その他の重み付きコードを、次の表に掲載しておきます。

１０進数	6-3-1-1	5-2-1-1	5-1-1-1-1
0	0000	0000	00000
1	0001	0001	00001
2	0011	0100	00011
3	0100	0110	00111
4	0101	0111	01111
5	0111	1000	10000
6	1000	1001	11000
7	1001	1100	11100
8	1011	1110	11110
9	1100	1111	11111

表 2.4　その他の重み付きコード

2.4.3　３増コード

このコードは１０進数に３を加えた後に、２進数に変換するという手順で作ったコードです。

このコードの対応表を次に示しておきます。このコードは、4 と 5 を境にして上下対称、かつ 0 と 1 とを入れ替えたコードとなっています。

１０進数	３増コード
0	0011
1	0100
2	0101
3	0110
4	0111
5	1000
6	1001
7	1010
8	1011
9	1100

表 2.5　３増コード

2.4.4 グレイコード：循環コード (cyclic code) の一種

循環コードは、隣接するコード間で 1 ビットだけ違いす。次にその変換表を掲載しておきます。

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

表 2.6 グレイコード

このコード表を見ますと隣り合ったコードの間は必ず 1 ビットずれていることが分かります。更に 0 – 15、1 – 14、2 – 13、3 – 12、4 – 11、5 – 10、6 – 9 それと 7 – 8 を見ると下 3 桁は同じ値となっています。最上位の桁は 0 と 1 が入れ替わっていることが分かると思います。

このグレーコードは次の図 2.1 に示すように、１０進数を一度２進数へと変換した後、隣り合う数字ごとに数字が違う場合は 1、同じ数字の場合は 0 と置くことによって得られます。（排他的論理和） 排他的論理は XOR という文字が使われ、次の章で説明します。

次の図 2.1 は、数値 $43_{(10)} = 101011_{(2)}$ をグレーコードへ変換する場合の例です。

$$43 = 0--1--0--1--0--1--1$$
$$\searrow\ \swarrow\ \searrow\ \swarrow\ \searrow\ \swarrow\ \searrow\ \swarrow\ \searrow\ \swarrow\ \searrow\ \swarrow$$
$$1--1--1--1--1--1--0$$

図 2.1　グレーコードの求め方

もちろんグレーコードから 2 進数に戻すことも出来ます。上記のコード $43_{(10)} = 101011_{(2)}$ を使ってグレーコードから 2 進数への変換を、次の図に示しておきます。

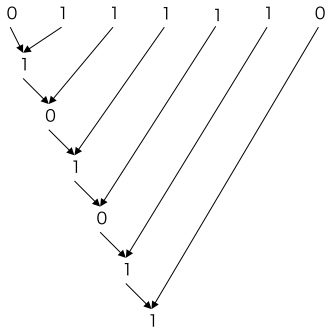


図 2.2　グレーコードから 2 進数への変換

この図のようにグレーコードの前に 0 を付け加えて、矢印の元に示してある数値の排他的論理和 NOR を取ることによって、元の 2 進数表示が得られます。

2.5 第 2 章問題

問題 1

次の 2 進数を 10 進数へ変換して下さい。

(a) 00001110 (b) 11100000 (c) 10000011 (d) 10011010

問題 2

次の 10 進数を 2 進数へ変換して下さい。

(a) 32 (b) 200 (c) 170 (d) 258

問題 3

次の 8 進数を 10 進数へ変換して下さい。

(a) 37 (b) 725 (c) 2476.2 (d) 1117.16

問題 4

次の 10 進数を 8 進数へ変換して下さい。

(a) 399 (b) 1500 (c) 600.5 (d) 3000.8125

問題 5

次の 8421BCD 数を 10 進数へ変換して下さい。

(a) 01011000 (b) 000100000000

(c) 1001.01110101 (d) 0011.0000011000100101

第 3 章

ディジタルの表現

ここではディジタル回路を始める前に必要となる二つの状態（値）のみを持つ集合論の基礎から話を始めます。集合論についての詳しい内容は、集合論の専門書を見て下さい。

本論に入る前に、使われる記号について話をしておく必要があります。まず始めに + と − の記号は次に示す意味を表します。この記号は、日本語で述べると、次のようになります。

> + :「・・・と・・・」あるいは「・・・および・・・」の意味を表します。論理和とも呼ばれます。
>
> $*$, $\cdot$, $\times$:「・・・あるいは・・・」の意味を表します。論理積とも呼ばれます。

以上のことから + と $*$,$\cdot$, $\times$（この著書では何も記号を付けないで表示します）の記号は、集合論では $\cup$, $\cap$ を用いて表現されます。

次に A , B , $\cdots$, X , Y , $\cdots$ 等のアルファベットは、0 または 1、ディジタル回路では True、faulse あるいは日本語で真、偽をあるいは ON、OFF などの二つの状態を示しています。

等号 = は、等号の左と右とが等しい、同じ価値があることを示しています。

次に 0 , 1 の意味ですが、通常は 0 は何もないことを、1 は全体を示しているのですが、2 進数を扱う場合 0 , 1 の二つしかありませんので、たまたま同じ数字の 1 となっているだけだと考えて下さい。

また定義、公理、定理という言葉が必要になりますが、これらは次のような意味を持っています。

> 定義：人が決めた取り決めですが、真理と矛盾してはなりません。
> 公理：誰も証明することが出来ない真理で、人類の経験則です。
> 定理：問題の答えの内、その他の問題にも使うことが出来る汎用性を持った答を示しています。

定理について、ある人は定理とはしないが、その他の人は定理としているものが存在しています。定理に関しては少々ややこしいのですが、しかし人名が付いている定理は、多くの人が定理として認めている内容であり、重要な定理として取り扱われています。

3.1　ブール代数の定義

ブール代数は、George Boole(1813 – 1864) によって始まりました。この代数は、元（要素）が二つしかない集合論の数学的裏づけとして構成されています。ブール代数は、次に示す公理を満足する集合として定義されています。ここで上付きの – は、補集合[*1]を意味しています。補集合とは、日本語で言えば「・・・でない」と言うことになり、元の値が 0 であれば 1 を、1 であれば 0 となることを示しています。

ブール代数において、次に示す式が成り立つものとします。殆どの式は直感的にも簡単に成立していることを理解できると思います。

[*1] 元の集合以外の集合のことです。

交換律

$$X+Y=Y+X \tag{3.1}$$
$$XY=YX \tag{3.2}$$

分配律

$$X+(YZ)=(X+Y)(X+Z) \tag{3.3}$$
$$X(Y+Z)=XY+XZ \tag{3.4}$$

同一律

$$X+0=X \tag{3.5}$$
$$X1=X \tag{3.6}$$

補元律

$$X+\bar{X}=1 \tag{3.7}$$
$$X\bar{X}=0 \tag{3.8}$$

ブール代数であるためには、上に述べた規則を満足すれば良いことになります。これらの規則がブール代数の基本法則です。

(3.1)〜(3.8) を用いますと、次の定理が求められます。証明は省略しますが証明の仕方としては、大まかには三通りあります。

1. 公理を用いて式を変形して証明。
2. 次に説明するベン図を用いて証明。ただしこの方法は変数が 3 個を越えると複雑になりすぎる欠点があります。
3. 真理値表を用いて証明。元々の変数は 0 と 1 の二つしか取らないので、表にしても大した大きさにはなりません。また変数がいくつあっても問題にはならないので、便利な方法です。

$$X+X=X \qquad XX=X \tag{3.9}$$
$$X+\bar{X}=1 \qquad X\bar{X}=0 \tag{3.10}$$
$$X+1=1 \qquad X0=0 \tag{3.11}$$
$$(X+Y)+Z=X+(Y+Z) \qquad (XY)Z=X(YZ) \tag{3.12}$$

これらの式は、覚えておくと役に立つ式です。
さらに知っておくと便利な、次の定理も成立します。

$$A+X=1 \quad \text{かつ} \quad AX=0 \quad \text{ならば} \quad X=\bar{A} \tag{3.13}$$
$$\bar{\bar{X}}=X \tag{3.14}$$
$$\bar{0}=1 \qquad \bar{1}=0 \tag{3.15}$$
$$X(X+Y)=X \qquad X+XY=X \tag{3.16}$$
$$X(\bar{X}+Y)=XY \qquad X+\bar{X}Y=X+Y \tag{3.17}$$
$$XY+XZ=X(Y+Z) \qquad (X+Y)(X+Z)=X+YZ \tag{3.18}$$
$$XY+YZ+\bar{X}Z=XY+\bar{X}Z \qquad (X+Y)(\bar{X}+Z)=XZ+\bar{X}Y \tag{3.19}$$
$$XY+\bar{X}Y=Y \qquad (X+Y)(\bar{X}+Y)=Y \tag{3.20}$$

その他に集合論で出てくる重要なド・モルガンの定理 が成り立ちます。

定理 1（ド・モルガンの定理）

$$\overline{X+Y}=\bar{X}\bar{Y} \tag{3.21}$$
$$\overline{ZY}=\bar{X}+\bar{Y} \tag{3.22}$$

この証明は X , Y に具体的に 0 ,$=1$ を代入して、計算すればすぐに求められますので主略します。

次にベン図と真理値表について説明しておきます。

3.2　図または表による表現

電子回路として実現する前に、単に式だけで表現するだけでは実現が難しいので、別の表現が必要になります。それがこれから説明するベン図と真理

値表です。

3.2.1　ベン図

ベン図はブール代数の図的表現です。その他の代数の場合も同じですが、図による表現は直感的に状況を捉えることが出来ますので非常に有効な手段ですが、細かいところまで表現することが難しいことはベン図の欠点です。

図 3.1 に 2 変数の場合のベン図を示しておきます。このベン図において、四角い枠の中は世界全体を意味しています。事象 A および事象 B は各々円で囲まれた中の領域で示され、論理積 AB はこれら二つの円が重なった領域、論理和 $A+B$ は二つの円によって囲まれた領域によって示されています。

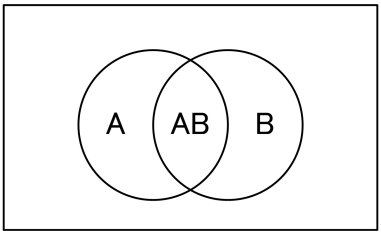


図 3.1　ベン図

この説明から分かりますように、変数が三つを越えると図が非常に複雑になってしまい、解釈することが困難になってきます。そのためベン図はせいぜい三つまでの変数に対する説明には何とか使えますが、それ以上になると使うことは殆どありません。

3.2.2　真理値表

真理値表は、ブール代数の表による表現です。ブール代数を表の形に纏めて記述できるのは、各変数が二つの値しか取らないためであり、このことが

実数などを扱う代数とは大きく異なる点です。

表の一番左は単に番号を示しており、場合によってこの番号は必要ないので、省略されます。2 番目から 4 番目の列は、各変数に対応した真理値を示し、一番右が式の結果を示しています。

No.	A	B	C	関数
0	0	0	0	$\bar{A}\bar{B}\bar{C}$
1	0	0	1	$\bar{A}\bar{B}C$
2	0	1	0	$\bar{A}B\bar{C}$
3	0	1	1	$\bar{A}BC$
4	1	0	0	$A\bar{B}\bar{C}$
5	1	0	1	$A\bar{B}C$
6	1	1	0	$AB\bar{C}$
7	1	1	1	ABC

表 3.1　真理値表の例

この場合には一番右の列には、各変数に対応した式を記入しておきましたが、実際の問題においては、問題に応じた 0 , 1 の値が入ることになります。

3.3　標準形

ブール代数を表現する方法として、標準形という概念があります。この標準形という形式は、これからディジタル論理回路に関する様々な計算を行うために必要な概念です。標準形は、ディジタル回路でしばしば出てきます。

標準形には二種類存在します。いわゆる Dual と言われる二つの形式であり、それぞれ加法標準形、 乗法標準形と呼ばれています。

加法標準形とは、XY の項を論理和で結合した形で表され、乗法標準形とは、$X+Y$ の項を論理積の形で結合した式で表されます。

例えば混合の形である $A(C+D)+B(C+D)$ という式を加法標準形、乗法

標準形で表しますと、次のような式となります。

$$AC+AD+BC+BD \qquad \text{加法標準形}$$
$$(A+B)(C+D) \qquad \text{乗法標準形}$$

ディジタル回路は 0 と 1 の二つしか取らない電子回路ですので、回路全体としては 0 と 1 の二つしか生じない部品の集合体であると考えられます。このことからブール代数は、ディジタル回路において重要な役割を果たすことになります。

3.4　ディジタル回路の回路表現

ディジタル回路を回路で表現するに当たり、基本的な素子がいくつかありますので、ここではそれらの素子について述べていきます。

これから示すディジタル回路の基本素子は簡単な記号で示されますが、その中身は当然電子回路で構成されています。しかしどのような電子回路が用いられているかということに関しては一切関知しません。極端な言い方をすれば各素子の機能を持ちさえすれば、何でも良いということです。

NOT

NOT についての記号と論理式とを、次の図に示します。

記号	論理式
X ─▷○─ Z	$Z=\bar{X}$

図 3.2　NOT の記号と論理式

次に NOT の真理値表を載せておきます。

X	Z
0	1
1	0

表 3.2　OR の真理値表

OR

OR についての記号と論理式とを、次の図に示します。

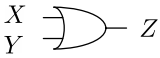
記号	論理式
	$Z = X + Y$

図 3.3　OR の記号と論理式

次に OR の真理値表を載せておきます。

X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	1

表 3.3　OR の真理値表

AND

AND についての記号と論理式とを、次の図に示します。

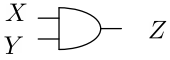
記号	論理式
	$Z = XY$

図 3.4　AND の記号と論理式

次に AND の真理値表を載せておきます。

X	Y	Z
0	0	0
0	1	0
1	0	0
1	1	1

表 3.4　AND の真理値表

NOR

NOR とは not OR のことを示しています。NOR についての記号と論理式とを、次の図に示します。

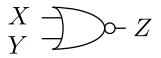
記号	論理式
	$Z = \overline{X + Y}$

図 3.5　NOR の記号と論理式

次に NOR の真理値表を載せておきます。

X	Y	Z
0	0	1
0	1	0
1	0	0
1	1	0

表 3.5　NOR の真理値表

NAND

NAND とは not AND のことを示しています。AND についての記号と論理式とを、次の図に示します。

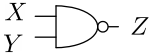
記号	論理式
	$Z = \overline{XY}$

図 3.6　NAND の記号と論理式

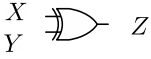
次に NAND の真理値表を載せておきます。

X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	1

表 3.6　NAND の真理値表

XOR

XOR は排他的論理和と呼ばれています。XOR についての記号と論理式とを、次の図に示します。

記号	論理式
	$Z = X \otimes Y$

ただし　$X \oplus Y = X\bar{Y} + \bar{X}Y$

図 3.7　XOR の記号と論理式

次に XOR の真理値表を載せておきます。

X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	0

表 3.7　XOR の真理値表

3.5　第３章問題

問題 1

ド・モルガンの二つの法則を証明して下さい。

$$\overline{A+B} = \bar{A}\bar{B} \tag{3.23}$$

$$\overline{AB} = \bar{A} + \bar{B} \tag{3.24}$$

問題 2

表に示す真理値表から、加法標準形および乗法標準形を求めて下さい。

入力	出力	入力	出力
A B C	Y	A B C	Y
0 0 0	0	1 0 0	0
0 0 1	0	1 0 1	0
0 1 0	0	1 1 0	1
0 1 1	1	1 1 1	0

表 3.8　問題 2

問題 3

次の式をブール代数を用いて簡単化して下さい。

$$X = A\bar{B}\bar{C} + AB\bar{C} + \bar{A}\bar{C} \tag{3.25}$$

$$Y = M\bar{N}P + \bar{L}P + \bar{L}M\bar{N} + \bar{L}MN\bar{P} + \bar{L}\bar{N}\bar{P} \tag{3.26}$$

$$Z = AB\bar{C}\bar{D} + \bar{A}B\bar{D} + \bar{A}\bar{B}C + \bar{B}D \tag{3.27}$$

問題 4

(a) 次の式を加法標準形に変換して下さい。

$$X = \overline{(A + \bar{B} + \bar{C} + D)(A + \bar{B} + \bar{C} + \bar{D})} \tag{3.28}$$

(b) 次の式を乗法標準形に変換して下さい。

$$Y = \overline{\bar{A}\bar{B}\bar{C}D + \bar{A}B\bar{C}\bar{D}} \tag{3.29}$$

問題 5

次の回路図を真理値表で表して下さい。

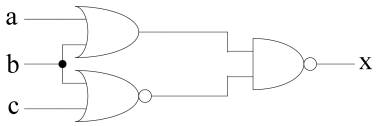


図 3.8　問題 5 の図

第 4 章

最少化とディジタル化

回路を実現する際に、出来うるならば最少の部品点数で回路を組み立てたいものです。それは

- 経費を節約する
- 回路を組み立てるための労力を削減したい
- 回路の信頼性が向上する

ということからも重要なことです。ここではその最少の部品点数で実現するための、数学的手法について記述することにします。

最少化の手法は、大きく分けると、次のような手法に分けることが出来ます。

- 代数計算
- ベン図
- 真理値表
- ヴァイチ図
- カルノー図
- クワインーマクラフスキー法

などです。これらの手法について、述べていくことにします。

このうち代数計算の手法は、最少化を行う手法としてはあまり適切ではありません。例えば最少化をするために $A\bar{B}+\bar{B}C+\bar{A}C$ が、最少化によって $A\bar{B}+\bar{A}C$ となる、ということなどを用いることは、かなりの代数計算の慣れが必要となるからです。

4.1　ベン図による最小化

ベン図は第 3 章にも示しましたように、次の図 4.1 によって与えられます。このベン図はせいぜい三変数までしか扱うことが出来ないため、用途も非常に限られます。ディジタル回路の解析や合成に用いられることはほとんどありませんので、ベン図についてはこれ以上話をしません。

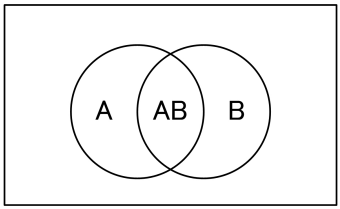


図 4.1　ベン図

4.2　真理値表による最小化

真理値表の例として、次の第 3 章で出てきた表 3.1 をもう一度考えてみます。

No.	A	B	C	関数
0	0	0	0	$\bar{A}\bar{B}\bar{C}$
1	0	0	1	$\bar{A}\bar{B}C$
2	0	1	0	$\bar{A}B\bar{C}$
3	0	1	1	$\bar{A}BC$
4	1	0	0	$A\bar{B}\bar{C}$
5	1	0	1	$A\bar{B}C$
6	1	0	0	$A\bar{B}\bar{C}$
7	1	1	1	ABC
8	1	1	0	$AB\bar{C}$

表 4.1 真理値表の例

この真理値表を見て最少化を考えるのですが、例えば *No* 2 と *No* 3 を考えてみますと同じ *A* と *B* の値ですが *C* だけが異なっています。この場合には *C* は冗長ですので、消去することが出来ます。式で書きますと、次のようになります。

$$A\bar{B}\bar{C} + A\bar{B}C = A\bar{B} \tag{4.1}$$

真理値表を用いて、最少化を考える場合には、このようにして式を簡単化して行います。結果的には代数の定理を用いて元の式を最少化していくことになります。

より具体的な例として、次のような真理値表 4.2 の場合について考えてみます。この表の中で出力結果が 1 となっている部分に注目して最少化を考えてみます。1 抱き絵を考えさえすれば、残りは全て 0 の我愛となりますので、1 の場合にどのような結果となるかということが分かりさえすれば良いのです。

出力結果が 1 の場合を考えなければならないということではありません。求める結果は真理値表を忠実に再現すればよいわけですので、結果が 0 となる場合のみを取り上げて考えてもよいのですが、この場合には 0 の数に比べて 1 となる場合の数の方が少ないため、たまたま結果が 1 となる場合を取り上げて考えているにすぎません。

最終結果が 1 となる場合ですので、論理代数の和を考えますと、最終結果の式の各項が 1 となればその他の項が何であろうとも最終結果が 1 となるので、真理値表における出力が 1 となる項の代数和を取ることによって最終結果が得られることが分かります。

先程述べたように結果が 0 となる場合を取り上げても特に問題がでるわけではありません。その場合には最終結果が 0 になればよいわけですので、その場合には論理積を考えればよいのです。つまり真理値表における結果の論理積を考えれば、各項が 0 となれば最終結果は必ず 0 となる、つまりその逆は必ず 1 ですからこの場合において元の真理値表を忠実に再現していることになるので、正しい結論が得られていることになります。ここでは一番右の列に式を追加しておきます。

No.	A	B	C	D	f	式
0	0	0	0	0	0	$\bar{A}\bar{B}\bar{C}\bar{D}$
1	0	0	0	1	0	$\bar{A}\bar{B}\bar{C}D$
2	0	0	1	0	0	$\bar{A}\bar{B}C\bar{D}$
3	0	0	1	1	0	$\bar{A}\bar{B}CD$
4	0	1	0	0	0	$\bar{A}B\bar{C}\bar{D}$
5	0	1	0	1	1	$\bar{A}B\bar{C}D$
6	0	1	1	0	0	$\bar{A}BC\bar{D}$
7	0	1	1	1	1	$\bar{A}BCD$
8	1	0	0	0	0	$A\bar{B}\bar{C}\bar{D}$
9	1	0	0	1	0	$A\bar{B}\bar{C}D$
10	1	0	1	0	0	$A\bar{B}C\bar{D}$
11	1	0	1	1	0	$A\bar{B}C$
12	1	0	1	0	0	$A\bar{B}\bar{C}\bar{D}$
13	1	1	0	1	1	$AB\bar{C}D$
14	1	1	1	0	0	$ABC\bar{D}$
15	1	1	1	1	1	$ABCD$

表 4.2　真理値表の具体例

この真理値表 4.2 から 1 になる項を抜き出して、簡略化しますと次のよう

な結果が得られます。

$$\begin{aligned} f &= \bar{A}B\bar{C}D + \bar{A}BCD + AB\bar{C}D + ABCD \\ &= \bar{A}BD + ABD \\ &= BD \end{aligned} \tag{4.2}$$

この例から分かりますように、真理値表を用いて最少化する方法は、かなり直感に頼らざるを得ません。しかも見落としが発生しやすい方法です。真理値表自体はその他の手法においても用いられますが、最少化を目的として用いられることはほとんどありません。

4.3　ヴァイチ図による最小化

これはベン図を改良した手法で、1952 年 E. W. Veitch によって考案されました。二変数と三変数の場合について次の図に示しておきます。

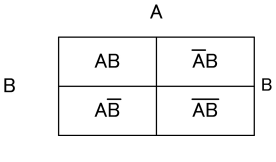


図 4.2　二変数のヴァイチ図

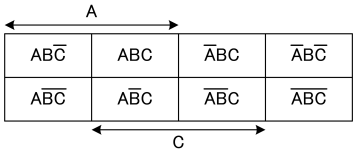


図 4.3　三変数のヴァイチ図

この図を見ると分かりますようにヴァイチ図では、隣接するセルの間ではただ 1 個の変数だけが異なるように並べられています。また各セルには変数の総ての組み合わせが、セルとして置かれています。このヴァイチ図はベン図を改良したものとして便利ではありますが、更に改良を加えられたカルノー図の方が、今日では用いられていますので、ヴァイチ図の説明はこれくらいにして、次にカルノー図についての説明にはいることにします。

4.4　カルノー図

最小化によって、最もよく用いられているカルノー図について述べています。

4.4.1　カルノー図による表現

この図は、1953 年に G. Karnaugh によって考案された手法であり、そのためカルノー図と呼ばれています。ヴァイチ図との違いは次のような点にあります。

1. 変数を示すアルファベットの配置を替えた
2. 各セルの中に変数を表す文字の代わりに、1 と 0 を入れた

カルノー図もヴァイチ図と同じように各隣接するセルの間では、変数の変化は一つです。このカルノー図の例として図 4.4 を示しておきます。

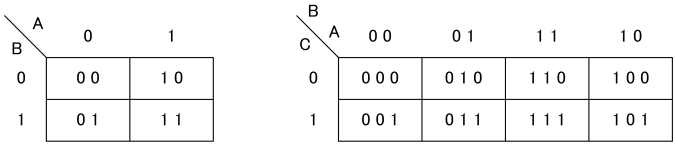


図 4.4　カルノー図の例

この例は二変数と三変数の場合についてのカルノー図ですが、五変数以上になると単純なカルノー図だけでは困難が生じます。そのような場合には別の工夫をする必要がでてきます。五変数の場合の二つの方法を図 4.5、4.6 に示しておきます。

C D E \ A B	0 0	0 1	1 1	1 0
0 0 0	00000	01000	11000	10000
0 0 1	00001	01001	11001	10001
0 1 1	00011	01011	11011	10011
0 1 0	00010	01010	11010	10010
1 1 0	00110	01110	11110	10110
1 1 1	00111	01111	11111	10111
1 0 1	11101	01101	11101	10101
1 0 0	00100	01100	11100	10100

図 4.5　五変数の例 *ABCDE*

E \ C D \ B A	0 0	0 1	1 1	1 0
0 0 0	00000	10000	11000	01000
0 0 1	00001	10001	11001	01001
0 1 1	00011	10011	11011	01011
0 1 0	00010	10010	11010	01010
1 1 0	00110	10110	11110	01110
1 1 1	11111	10111	11111	01111
1 0 1	00101	10101	11101	01101
1 0 0	00100	10100	11100	01100

図 4.6　五変数の例 *EDCBA*

4.4.2　重み付け

カルノー図を取り扱うとき、各セルに対して重み付けをしておくと便利な場合があります。この重み付けは、例えば $A = 2^0$, $B = 2^1$, $C = 2^2$, $\cdots$

のように行います。この重み付けの例を、図 4.7〜図 4.9 に示しておきます。

B / C	A 00	01	11	10
0	0	1	3	2
1	4	5	7	6

図 4.7　重み付きカルノー図 (a)

B / C	A 00	01	11	10
00	0	1	3	2
10	4	5	7	6
11	12	13	15	14
10	8	9	11	10

図 4.8　重み付きカルノー図 (b)

E D C / B A	00	01	11	10
000	0	1	3	2
001	4	5	7	6
011	12	13	15	14
010	8	9	11	10
110	24	25	27	26
111	28	29	31	30
101	20	21	23	22
100	16	17	19	18

図 4.9　重み付きカルノー図 (c)

4.4.3　カルノー図による最小化

カルノー図を用いた最少化について考えていきます。まずはじめにカルノー図の隣接するセルの間では、必ず 1 個の変数だけが違うように構成されていました。また図の端の方にあるセルと、ちょうどその反対側にあるセルとの間の変数の個数も 1 個違うだけです。よって隣接するセルが同じ真あ

るいは偽であるならば、それらを一纏めにすることによって変数の数を減らすことが出来ます。このことを例を考えることによって示していきます。

カルノー図の中に示されるのは、ある問題を実行したときに、いくつかの変数が存在しそれらの変数が真かあるいか偽かによって出力の結果の真偽が決まるわけですので、カルノー図の中にその結果を真つまり 1 を、偽 0 を記載していきます。

次の式で示す、具体的な例を述べていきます。

$$f = \bar{A}B\bar{C} + \bar{A}\bar{C}D + \bar{A}\bar{B}\bar{D} + AC + BC\bar{D} + \bar{A}\bar{C}D \tag{4.3}$$

この式は図 4.10 のカルノー図を与えます。このカルノー図を表す簡単化した式を求めることが目的です。まずはじめにこの式が図 4.11 のカルノー図で与えられることは、$ABCD$ の各々に 1 と 0 を代入しても求められますが, この方法だと $2\times2\times2\times2 = 16$ 通りについて計算しなければなりません。

この方法よりも元の式を四つの変数に変形して表 4.2 を利用することによって確かめる方が楽だと思います。そのために $X + X = X$ と $1 = X + \bar{X}$ という関係式を用います。例えば $\bar{A}B\bar{C} = \bar{A}B\bar{C}(D + \bar{D}) = \bar{A}B\bar{C}D + \bar{A}B\bar{C}\bar{D}$ このように変形する全ての項を四つの変数で表現して、表 4.2 から 5 と 4 が得られ、そこから各々 0110 と 0101 のところに 1 が入ることになります。これを順次適用していくとカルノー図 4.11 が得られます。

カルノー図において 1 となるセルについて考えます。図 4.11 のように 1 を囲んでみますと、囲まれた 1 において式の変数の値が逆転する、つまり例えば X と $\bar{X}$ となる場合は、変数が無くなることになります。この性質を利用しますと、その結果は (4.4) 式のように与えられます。

$$f = \bar{A}\bar{C} + \bar{A}\bar{D} + AC \tag{4.4}$$

別の囲み方をした場合の最少化の例と式とを、図 4.12、(4.5) 式に示しておきます。

$$f = \bar{A}\bar{C} + C\bar{D} + AC \tag{4.5}$$

(4.4) 式と (4.5) 式とは、一見違う結果のようにみえますが、同じ結果を導く別の表現です。

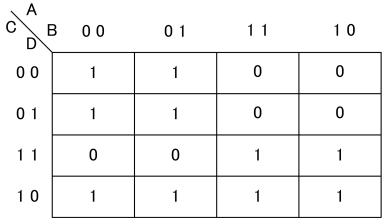


図 4.10　カルノー図最少化の例

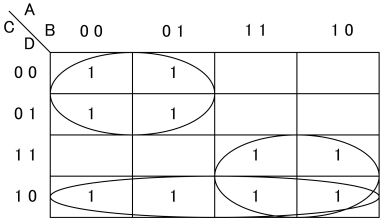


図 4.11　最少化（1）

また 0 を基準にしても最少化は可能でです。図 4.10 において 0 のところをた 1 と見なしますと $A\bar{C}+A\bar{C}D$ となりますが、これが 0 となるにはこの式全体を 0 へ変換すれば関数 f 全体を示していることになります。その例と式とを、図 4.13、(4.7) 式に示しておきます。

$$f = \overline{A\bar{C}+A\bar{C}D} \tag{4.6}$$
$$= (\bar{A}+C)(\bar{A}+C+\bar{D}) \tag{4.7}$$

この場合の式は、和の積という形になります。0 のブロックの中で 1 と 0 が現れる変数は消去して、1 となる変数はその反対を取ることになります。更に各ブロックの間は積でつないでいかねばなりません。

このようなカルノー図のように 1 の数よりも 0 の数が少ない場合には、最後の例のように 0 を考えた方が楽です。

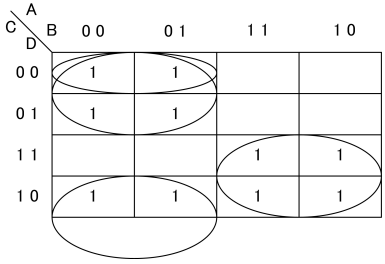


図 4.12　最少化（２）

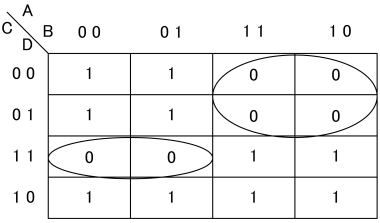


図 4.13　0 を取った場合の最少化

この結果を含めて四種類の違う結果が得られましたが、全て同じ結論を導く別の表現ということになります。

4.4.4　カルノー図の多段化

今までの例からも分かりますように、カルノー図は変数の数が 4 を越えると取り扱い方が難しくなってきます。つまりカルノー図を考える限り取り扱える変数の数は、以前にも述べましたようにせいぜい 5 個までです。変数が 4 個を越える場合には、特別な工夫をしなければなりません。その工夫として各変数について場合分けをする方法が考えられます。そのいくつかの手法について述べていくことにします。

場合分けをして求める方法

例として四変数の場合について考えてみます。A , B , C , D の四変数を考えます。場合分けを考えるのですが、A の値が 0 と 1 の場合を考えなければなりませんが、残りの変数と A との関係は積を考え、A の値が違う場合の和を取らなければならないので、A の値が 0 となる場合は、和をとっても関係が無くなりますので考える必要がありません。そこで図 4.14 の場合だけとなります。この図の中でセルの中の数値は、$ABCD$ の順に並べたときの１０進数を示しています。

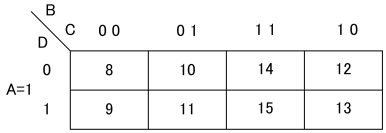


図 4.14　四変数の最少化の例

この図を用いてカルノー図が図 4.15 のように与えられた場合を考えます。このときのカルノー図による最少化された図と式は、図 4.16 と (4.8) 式として得られます。

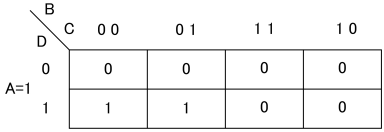


図 4.15　４変数の最少化例

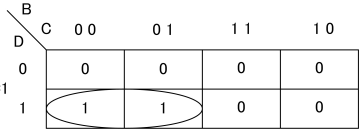


図 4.16　四変数の最少化の結果

$$
\begin{aligned}
f &= A(\bar{B}D) \\
&= A\bar{B}D
\end{aligned} \tag{4.8}
$$

変数を別のセルに含ませる方法

$f = \sum(4, 5, 6, 7, 8, 9, 10)$ の場合に 1 となる関数を考えます。数字は A , B , C , D の四つの変数で表した 2 進数の１０進数表示を表しています。変数 D に注目して、D が含まれないカルノー図を描くと図 4.17 のように与えられます。このセルの中の数値は１０進数を示しています。

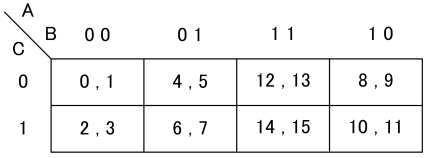


図 4.17　変数 D を含まないカルノー図

このセルの中で 11 のとき $f=0$ ですので、図 4.18 のように場合分けをしなければなりません。

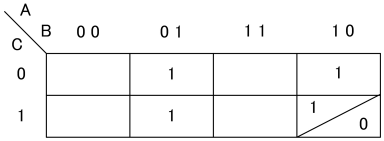


図 4.18　場合分けをしたカルノー図

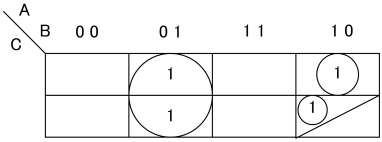


図 4.19　場合分けをしたカルノー図の最少表現を求める図

図 4.19 から求める加法標準形は、(4.9) 式のようになります。[*1]

$$\begin{aligned} f &= \bar{A}B + A\bar{B}\bar{C} + A\bar{B}C\bar{D} \\ &= \bar{A}B + A\bar{B}(\bar{C} + C\bar{D}) \\ &= \bar{A}B + A\bar{B}\bar{C} + A\bar{B}\bar{D} \end{aligned} \tag{4.9}$$

元の式を分解する方法

同じ例について考えます。次のように式を分解します。

$$f = \sum(4\,,\,5\,,\,6\,,\,7) + \sum(8\,,\,9\,,\,10)$$

このように分解したのは、右辺の第 1 項は $A=0$、第 2 項は、$A=1$ となるように行っています。こうすることにより、元の式は二つのカルノー図に

[*1] ここで第 3 章 (3.17) 式を用いていることに注意。

よって与えられ、各々のカルノー図から求められた最少表現の論理和を求めることによって、元の式の最少表現が求められることになります。各々のカルノー図 4.20、4.21 と最少表現を図 4.22、4.23 に示しておきます。

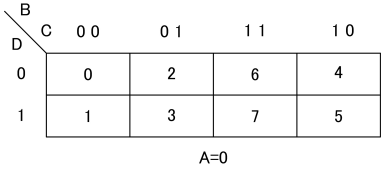


図 4.20　第 1 項のカルノー図

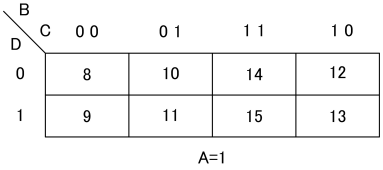


図 4.21　第 2 項のカルノー図

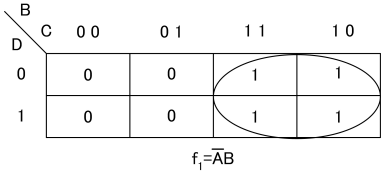


図 4.22　第 1 項の最少表現

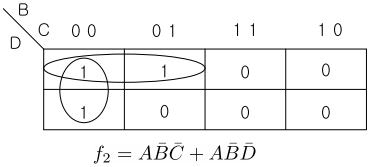


図 4.23　第 2 項の最少表現

4.5　クワイン・マクラスキー法による最小化

クワイン・マクラスキー法 (Quine-McClusky method：ＱＭ法) は、多変数関数を最少化する場合ほとんど機械的に取り扱うことが出来るため、計算機を用いて最少化する場合によく用いられます。この方法も $XY + \bar{X}Y = Y$ を繰り返し用いて最少化が行われます。クワイン・マクラスキー法は、次のような手順によって行われます。

1. 加法標準形に直す。
2. 各項に対応する 10 進数を総て表にする。

3. 2 進表現の 1 の個数が同じ項毎に分類する。
4. 各項毎にその下にある項と比較し一箇所だけ異なる項を別の表に書き出す。このとき ∗ 点をつける。これを主項と名付けます。
5. これを繰り返し主項を求める。
6. 主項の表を作成し、必須主項（必須項）を見つけ出す。
7. 必要な表現式を得るための、もっとも適切と思われる主項を選ぶ。

この手順を、例を示して説明していくことにします。次の式によって示される式の最少化を考えてみます。

$$f = \bar{A}B\bar{C} + \bar{A}BD + A\bar{B}\bar{C} + A\bar{B}D + \bar{A}BC \tag{4.10}$$

先ほどの手順（1，2，3）に従って、次の表 4.3 に示す表を作成します。ただし同じ項は省略しています。

標準項	２進表現	１０進表現
$\bar{A}B\bar{C}D$	0 1 0 1	(5)
$\bar{A}B\bar{C}\bar{D}$	0 1 0 0	(4)
$\bar{A}BCD$	0 1 1 1	(7)
$A\bar{B}\bar{C}D$	1 0 0 1	(9)
$A\bar{B}\bar{C}\bar{D}$	1 0 0 0	(8)
$A\bar{B}CD$	1 0 1 1	(11)
$\bar{A}BC\bar{D}$	0 1 1 0	(6)

表 4.3　ＱＭ法の例

この表から手順（4）に従って、次の表 4.4 を作成します。この表の中で、1 の値がどの項とも一つ異なることが無い項を主項として ∗ を付けて残します。

代数表現	2 進表現	10 進表現
$\bar{A}B\bar{C}$	0 1 0 −	(4 , 5)
$\bar{A}BD$	0 1 − 1	(5 , 7)
$\bar{A}B\bar{D}$	0 1 − 0	(4 , 6)
$\bar{A}BC$	0 1 1 −	(6 , 7)
$A\bar{B}D^*$	1 0 − 1	(9 , 11)
$A\bar{B}\bar{C}^*$	1 0 0 −	(8 , 9)

表 4.4　表 4.3 の簡略化

表 4.4 の中で、一番右の項は、表 4.3 で与えられる項の 10 進数表現を示しています。同じような手順を繰り返して、次の表 4.5 が得られます。

代数表現	2 進表現	10 進表現
$\bar{A}B^*$	0 1 − −	(4 , 5 , 6 , 7)

表 4.5　2 度目の簡略化

以上の結果から主項を求めるのですが、そのために 表 4.4 と表 4.5 から次のような表を作ります。この表は主項表と呼ばれており、その他の項と比較して消去することが出来なかった項から選ばれています。この項は、表 4.4、表 4.5 において ∗ の印が付けられている項です。

		標準項						
		$\bar{A}B\bar{C}D$	$\bar{A}B\bar{C}\bar{D}$	$\bar{A}BCD$	$A\bar{B}\bar{C}D$	$A\bar{B}\bar{C}\bar{D}$	$ABCD$	$\bar{A}BC\bar{D}$
		0101	0100	0111	1001	1000	1011	0110
主項		(5)	(4)	(7)	(9)	(8)	(11)	(6)
(9 , 11)	$A\bar{B}D$				×		⊗	
(8 9)	$A\bar{B}\bar{C}$				×	⊗		
(4 , 5 , 6 , 7)	$\bar{A}B$	⊗	⊗	⊗				⊗

$f = \bar{A}B + A\bar{B}\bar{C} + A\bar{B}D$

表 4.6　最終結果

この表の中で、一番下に記載した方程式が、最終的に得られた式です。表 4.6 において × 印は、主項において含まれる標準項であることを示していま

す。また ◯ 印で囲んであるのは、その行の主項が必須項であることを示しています。

よって最終的に得られた方程式は、総ての標準項 を含むのに必要な必須項の和として求められています。

4.6 ディジタル化

ディジタル回路の中は、当然のようにディジタル信号のみしか扱っていないのですが、現実社会はアナログの世界です。その現実の社会をディジタル信号として表現できなければ、ディジタル回路は全く意味が無くなってしまいます。アナログの信号をディジタルへと変換する、アナログーディジタル変換器（ＡＤ変換器）や、その逆の動作を行うディジタルーアナログ変換器（ＤＡ変換器）については、後の第 14 章で述べることにして、現実の現象をそのままディジタルで表現できる場合がありますので、ここではその場合について考えてみることにします。

いくつかの例について考えてみることにします。

例 1

3 個の ON,OFF スイッチが論理回路の入力として接続されているとします。出力端子は 1 個です。出力にはランプが接続されており、スイッチが 1 個または 3 個が ON のときランプが点灯し、それ以外のときは消灯するようにします。

1. 真理値表を求めで下さい。
2. カルノー図を求めて下さい。

入力を A , B , C として、真理値表は次のように簡単に求まります。

A	B	C	出力
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

表 4.7　スイッチの真理値表

カルノー図は上記真理値表から、次のように与えられます。

A / B → / C ↓	0 0	0 1	1 1	1 0
0	1	1	0	1
1	1	0	1	0

図 4.24　例 1 のカルノー図

例 2

0 から 9 までの１０進数を 8421BCD コードで表した場合について考えていきます。ただし 10 以上の数値は、無いものとします。これはパソコンのキーボードのテンキーに相当します。テンキーには 10 以上のキーは存在していません。つまり 10 以上の数値に対しては、どの様な結果が出ようとも何ら問題にはなりません。この様な場合をドントケア (Don’t Care)（どうでも良い）と呼んでいます。

0 から 9 までの１０進数を 8421BCD コードへの変換回路は、出来ているとします。3 の倍数の１０進数のとき、出力が発生するような回路を作成したいとします。

1. 真理値表を求めて下さい。
2. カルノー図を求めて下さい。

真理値表は、次のように与えられます。

A	B	C	D	出力
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1

表 4.8　テンキーの真理値表

この真理値表から、次のカルノー図が得られます。

C＼A B＼D	00	01	11	10
00	0	0	−	0
01	0	0	−	1
11	1	0	−	−
10	0	1	−	−

図 4.25　例 2 のカルノー図

この中で − はドントケアを示しています。

4.7　問題

問題 1

次の真理値表の加法標準形を求めて下さい。

入力	出力	入力	出力
A B C D	Y	A B C D	Y
0 0 0 0	1	1 0 0 0	1
0 0 0 1	0	1 0 0 1	0
0 0 1 0	1	1 0 1 0	1
0 0 1 1	0	1 0 1 1	0
0 1 0 0	0	1 1 0 0	1
0 1 0 1	0	1 1 0 1	0
0 1 1 0	1	1 1 1 0	1
0 1 1 1	0	1 1 1 1	0

表 4.9　問題 1

問題 2

問題 1 の式からカルノー図を求め、最小化のためのループを描いて下さい。

問題 3

問題 2 の結果から、最小の加法標準形を求めて下さい。

問題 4

問題 1 の表を用いて、乗法標準形を求めて下さい。

問題 5

問題 4 の結果を用いて、カルノー図を描き、それを用いて最小の乗法標準形を求めて下さい。

第 5 章

ディジタル回路素子

アナログ回路の場合の回路素子は、抵抗、容量、コイルなどの素子ですが、ディジタル回路の場合には、そのような素子まで遡らず、それらを組み合わせたセルという形での素子が基準となります。

このセルは、大きく二つの種類に分けることが出来ます。一つは組み合わせ回路と呼ばれ、もう一つは順序回路と呼ばれています。ここではこれらのセルについて説明していくことにします。

ディジタル回路の設計において、出発点は AND,OR,,NOT,NAND,NOR です。その中身については、このテキストでは扱いません。

ファンインとファンアウトという言葉があります。これは入力側あるいは出力側にいくつの入力数まであるいはいくつの出力数まで接続することが出来るかという数字を示しています。あまり多くの入力あるいは、出力を接続しますと電圧や電流の値が変化してしまい、正常に動作しなくなるためです。この数は、当然内部回路の形式に依存します。通常セルの中身を設計した部署からセルの供給と共に与えられます。

5.1　組み合わせ回路

組み合わせ回路には、いくつかの回路が考案されていますが、それらの回路について詳しく述べていきます。NOT、OR、AND、NOR、NAND、XOR については 3 章で記号と式それと真理値表を述べましたので、その内容以外のことについて記載しています。

5.1.1　OR と AND について

ド・モルガンの定理 によると次の関係が成立します。

$$\overline{\overline{A+B}} = \overline{A}\overline{B} \tag{5.1}$$

$$\overline{\overline{A}\overline{B}} = \overline{A}+\overline{B} \tag{5.2}$$

この式から分かることは、OR は AND と NOT があれば構成することが出来ることです。逆に AND は OR と NOT があれば構成することが出来ると言うことが分かります。このことを図で示すと図 5.1、図 5.2 のように示すことが出来ます。論理式は OR と AND によって表示されますので、NOT と OR あるいは AND の回路のいずれかを用いさえすれば、総てのディジタル回路は実現することが可能ということになります。

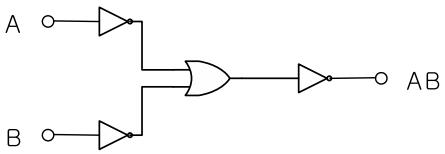


図 5.1　OR と NOT を用いた AND 回路

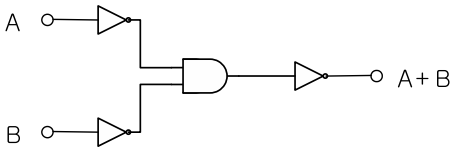


図 5.2 AND と NOT を用いた OR 回路

NOR と NAND について

NOR あるいは NAND 回路を用いて、NOT、OR、AND を実現することが出来ます。このことを NAND を使って、まず式で示してみますと、次のように証明することが出来ます。まず NOR の場合は

$$\overline{A+A} = \bar{A} \tag{5.3}$$
$$\overline{\bar{A}+\bar{B}} = \bar{A}\bar{B} \tag{5.4}$$
$$\overline{\overline{A+B}} = A+B \tag{5.5}$$

NAD の場合は、次の式が成立します。

$$\overline{AA} = \bar{A} \tag{5.6}$$
$$\overline{\bar{A}\bar{B}} = A+B \tag{5.7}$$
$$\overline{\overline{AB}} = AB \tag{5.8}$$

これらの (5.6)(5.7)(5.8) 式を回路図で示すと、図 5.3 から図 5.5 のように与えられます。

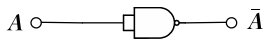


図 5.3 NAND を用いた NOT 回路

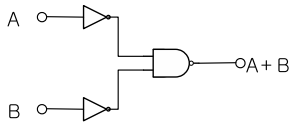


図 5.4　NAND を用いた OR 回路

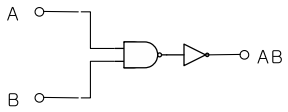


図 5.5　NAND を用いた AND 回路

5.2　フリップフロップ

順序回路は、回路の中に帰還作用を持たせた回路のことを一般的に称しています。帰還回路は、過去の情報を入力側へ持ってくることを意味していますので、いわゆる記憶装置が含まれていると言うことをも意味しています。そのためその動作は非常に複雑となりますが回路構成要素としては、組み合わせ回路のセルをお互いに接続して実現することが出来ます。

順序回路は、いくつかのフリップフロップと組み合わせ回路とを用いて実現されます。よって順序回路を理解するためには、その前にフリップフロップの知識が必要になってきます。ここではフリップフロップとしてどのような回路があるか、どのような動作をするかについて述べています。またその動作を説明するためのいくつかの手法についても述べることとします。

本論にはいる前に、いくつかの約束事がありますので、その説明を行います。

正論理、負論理：　正論理とは、電圧が高い場合を 1（あるいは ON、真）と定めて議論を行います。負論理とは、電圧が高い場合を 0（あるいは OFF、偽）と定めて議論を行うことを指しています。通常正論理がよく用いられます。

特性表：　入力状態と出力状態を表にしたものです。

励振表：　出力の変化に対して、そのときの入力の状態を示した表のことです。

Q^n：　現時点の状態を示しています。

Q^{n+1}：　クロックが入った後の状態を示しています。

CP：　クロック・パルスの略です。

Dフリップフロップ

Dフリップフロップは、Delay flip flop と呼ばれ、その名の示すようにデータ入力は、クロックパルス 1 個の時間だけ遅れて出力に現れます。そのブロック図と特性表及び励振表は、図 5.6、表 5.1、表 5.2 のように表されます。

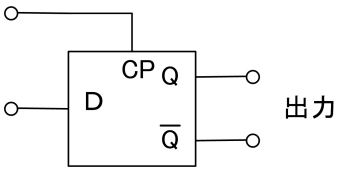


図 5.6　Dフリップフロップ

D	Q^{n+1}
0	0
1	1

表 5.1　D-FF 特性表

$Q^n \longrightarrow$	Q^{n+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

表 5.2　D-FF 励振表

Ｔフリップフロップ

Ｔフリップフロップは、toggle flip-flop と呼ばれ、コントロール入力Ｔが 1 の場合クロックパルスがくるたびに出力の状態を変えるフリップフロップのことを示しています。このフリップフロップのブロック図と特性表及び励振表は、図 5.3、表 5.4、表 5.5 のように表されます。

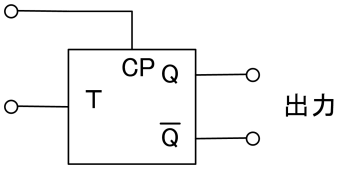


表 5.3　Ｔフリップフロップ

T	Q^{n+1}
0	Q^n
1	$\overline{Q^n}$

表 5.4　T-FF 特性表

$Q^n \longrightarrow$	Q^{n+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

表 5.5　T-FF 励振表

ＳＲフリップフロップ

SR フリップフロップは、set-reset flip-flop と呼ばれます。ＳＲフリップフロップには二種類あり、通常のＳＲフリップフロップ（ＳＲラッチと呼ばれることもあります）以外に同期式ＳＲフリップフロップと言われるフリップフロップがあります。 同期式ＳＲフリップフロップは、この後で述べることにして、ここではまず始めに通常のＳＲフリップフロップについて述べることにします。

通常のＳＲフリップフロップとして、二つの NAND を用いた回路を図 5.7 に示しておきます。このＳＲフリップフロップの特性表を表 5.6 に示しておきます。この特性表において、入力が共にゼロの場合は禁止状態と呼ばれ、このときの出力はどの様になるか分かりません。二つの入力が共に 1 の場合には、入力が変化する前の状態がそのまま保たれます。

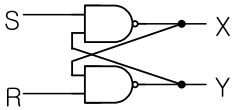


図 5.7　NAND を用いたＳＲフリップフロップ

S	R	Q^{n+1}	$\overline{Q^{n+1}}$
0	0	?	?
0	1	1	1
1	0	0	0
1	1	Q^n	$\overline{Q^n}$

表 5.6　SR-FF 特性表

同期式 SR フリップフロップ

同期式ＳＲフリップフロップは、ＳＲラッチに同期動作を付け加えたフリップフロップです。よって同期式ＳＲフリップフロップは、同期信号に合わせて動作します。例えば図 5.7 の回路を同期式ＳＲフリップフロップとするためには、図 5.8 のように回路を加え合わせることによって得られます。

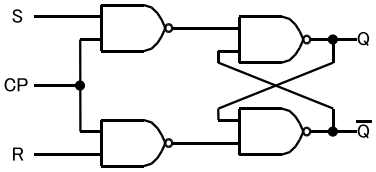


図 5.8　図 5.7 の同期化

ＳＲフリップフロップの動作を表 5.7 の特性表および表 5.8 の励振表に示しておきます。このフリップフロップの特性表を見て分かりますように、？の記号が入っています。これは入力側がともに 1 のときクロックが入ってくると、出力として 0 となるか 1 となるか予想がつかなくなることを意味しています。よってこの状態は禁止状態と呼ばれ、避けなければならない状態です。

励振表の中に ϕ という記号が入っているのは、don't care と呼ばれ、0 でも 1 でもかまわないことを意味しています。よって回路の最小化を行う際に、どちらか都合の良い状態を選ぶことが出来ます。

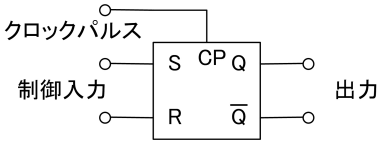


図 5.9　SR フリップフロップ

S	R	Q^{n+1}
0	0	Q^n
0	1	0
1	0	1
1	1	?

表 5.7　特性表

$Q^n \longrightarrow$	Q^{n+1}	S	R
0	0	0	ϕ
0	1	1	0
1	0	0	1
1	1	ϕ	0

表 5.8　励振表

ＪＫフリップフロップ

このフリップフロップは、ＳＲフリップフロップを改良したものと考えることが出来ます。つまりＳＲフリップフロップでは禁止状態がありましたが、ＪＫフリップフロップでは禁止状態は存在しません。

また表 5.10 の特性表から分かりますように、ＪとＫを一緒にするとＴフリップフロップとなります。またＪに入る信号を反転させてＫに入力することによりＤフリップフロップが得られます。さらに SR フリップフロップの特性表と比較すると分かりますように、SR フリップフロップをも含んでいることが分かります。以上のことからＪＫフリップフロップは、万能フリップフロップと見なすことが出来ます。

ＪＫフリップフロップのブロック図 5.10、特性表 5.9 および励振表 5.10 を次に示しておきます。

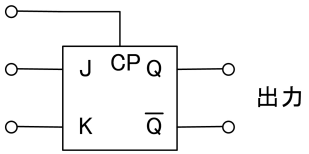


図 5.10　JK フリップフロップ

J	K	Q^{n+1}
0	0	Q^n
0	1	0
1	0	1
1	1	$\overline{Q^n}$

表 5.9　特性表

$Q^n \longrightarrow Q^{n+1}$		J	K
0	0	0	ϕ
0	1	1	ϕ
1	0	ϕ	1
1	1	ϕ	0

表 5.10　励振表

5.3　クロックパルス

ここでクロックパルス（CP or CK）について考察しておきます。

フリップフロップは、大部分クロックパルスに同期して動作しますが、クロックパルスの入力端子を持たないフリップフロップも存在します。先の例で示したラッチ回路などは、その例です。

クロックパルスを使うと言っても、同期式と非同期式論理回路が存在するように、一つのクロックだけで全ての論理回路の動作を行わせる同期式論理回路、数種類の位相がずれたパルスを用いて論理回路の動作を行わせる非同期式論理回路が存在します。

非同期式論理回路は一般的に解析が難しく、この著作においては取り扱っ

ていません。

クロックパルスに同期すると言っても、パルスのどの様な状態で回路が動作するかという問題が存在します。この場合、次のような同期方式が存在しています。

パルストリガ方式

この方式はマスタスレーブ型 とも呼ばれ、データを入力から出力へ転送するために、1 パルスの全体を必要とする方式です。ＪＫフリップフロップの大部分は、この方式を使っています。

トリガ方式

この方式は、パルスの立ち上がり部分もしくは立ち下がりの部分において、データを入力から出力へ転送する方式です。

立ち上がりトリガー方式フリップフロップ　パルスの立ち上がり部分（電圧が低いところから高いところへ移動する時間）で、フリップフロップの入力から出力へ信号が転送される方式です。

立ち下がりトリガー方式フリップフロップ　パルスの立ち下がり部分（電圧が高いところから低いところへ移動する時間）で、フリップフロップの入力から出力へ信号が転送される方式です。

以上の説明の中で、マスタスレーブ型ＪＫフリップフロップの動作について詳しく考えてみることにします。まず始めにＪＫの値に応じて、特性表において表 5.11 のような名前が付けられています。

J	K	Q	$\bar{Q}$	動作モード
0	0	Q_n	$\overline{Q_n}$	ホールド
0	1	0	1	リセット
1	0	1	0	セット
1	1	$\overline{Q_n}$	Q_n	トグル

表 5.11　マスタスレーブ型ＪＫフリップフロップの動作名称

マスタスレーブ型ＪＫフリップフロップの動作を理解するために、フリップフロップにパルスが入力したとき、どの様に動作するかと言うことを図 5.11 を用いて説明を進めます。

パルスの各部分において、番号を付けていますが、この番号の時間においてマスタスレーブ型ＪＫフリップフロップが、どの様な特性を持っているかということについて図の下の方に説明をしておきました。

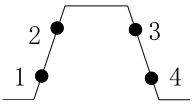


図 5.11　パルス期間での動作

1. ＪＫフリップフロップの入力と出力とが切り離されます。
2. ＪＫ端子に信号が入力されるが、出力には影響を与えません。
3. ＪＫフリップフロップの入力と出力とが切り離されます。
4. 入力の値に応じた出力が現れます。

この動作状態の中で問題なのは、クロックパルスが high のときです。クロックパルスが high のとき、ＪＫ入力のいずれの場合もこの期間の間に high の状態に一度でもなると、ＪＫフリップフロップはその入力を high と判断します。例えばクロックが high のとき、Ｊ入力あるいはＫ入力が共に、時間がずれていたとしても瞬間的に high 状態になったとしますとＪＫフリップフロップは、ＪＫ両入力が high であると判断し、トグル状態となり

ます。またいずれかの入力の一方が瞬間にでも high となると、ＪＫフリップフロップは、セットあるいはリセット状態であると判断してしまいます。

第 6 章

組み合わせ論理回路

組み合わせ論理回路とは、回路の中に帰還を含んでおらず、フリップフロップを使わずとも回路を実現できます。その分実現できる回路には限界があります。しかし回路解析、合成とも順序回路に較べて易しく、容易に実現することが出来ます。ここでは、組み合わせ回路の解析と合成に関して述べることにします。

いかなる場合も同じ様な感触ですが、解析の場合、筋道は多岐にわたっても結論は一つのものが得られる場合が多いのですが、合成の場合は、筋道も多岐にわたり、結論も様々なものが得られる場合が多いようです。そのぶん回路解析は比較的易しいのですが、それに較べて回路合成は、一般的に難しい問題を生じます。

6.1　組み合わせ回路の解析

組み合わせ論理回路の解析の目的は、任意の入力に対してどのような出力が得られるかを求めることです。よって入出力に関する真理値表が出来上がることにより解析は終了したことになります。

組み合わせ論理回路の解析は、比較的易しい問題です。回路が与えられたら、各ブロックについてブール代数式を求め、次に全体の回路について真理

値表を求めることによって得られます。

次の回路例について解析を行ってみます。

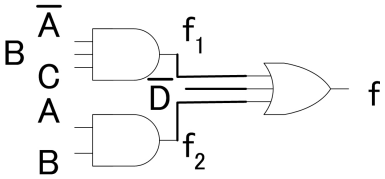


図 6.1　組み合わせ回路解析例

この回路は非常に簡単な回路ではありますが、たとえ複雑な回路であっても手法は同じ手法を用いることが出来ます。まずはじめに小さなブロックの式を求め、それらの式を結びつける式を次に求めていきます。回路図 6.1 の中に f_1 , f_2 , f と記載してあるのがそれです。このように記号を定めることによって、次のような式を簡単に求めることが出来ます。

$$f_1 = \bar{A}BC$$
$$f_2 = AB$$
$$f = f_1 + f_2 + \bar{D}$$

NAND 回路の解析

NAND 回路の解析式から、NAND には次の性質が存在します。

NAND 回路だけでで構成されている論理回路の場合には、特殊な回路解析手法を用いることが出来ます。

1. 出力から数えて奇数番目の NAND ゲートは、OR ゲートと考えられる。ただし変数は反転する。
2. 出力から数えて偶数番目の NAND ゲートは AND ゲートと考えられる。

このことを証明するために、最初のゲートに入る入力を A , B とします。

一段目の出力 二段目のあとの NAND 出力は $\overline{AB} = \bar{A} + \bar{B}$ となります。ここで A , B に注目しますと、AND になっており、しかも反転しています。

二段目の出力 次の NAND の入力として、上で求めた式と新しい入力変数として C を考えますと、その出力として $\overline{(\bar{A}+\bar{B})C} = (AB) + \bar{C}$ が得られます。ここで A , B に注目しますと、AND になっています。これは出力から数えて偶数番目の入力は最終出力で AND となっていることになっており、証明が出来ました。

三段目の出力 三段目の出力はもう一つの未知の入力を D とします。そうすると三つ段目の出力は $\overline{(AB) + \bar{C}D} = (\bar{A} + \bar{B}) + C) + \bar{D}$ となります。

以上の結果から、先程の二つの規則が出てきます。

例を図 6.2 に示します。

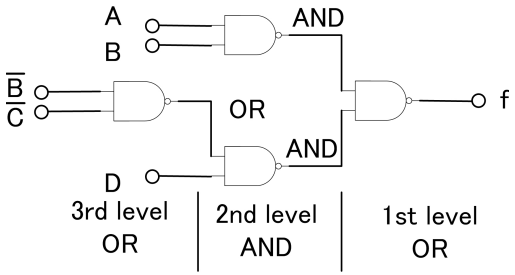


図 6.2 NAND 回路の例：$f = AB + D(B + C)$

NOR 回路の解析

NOR 回路で構成されている論理回路の場合には、特殊な回路解析手法を用いることが出来ます。NOR 回路は、次に示すような性質を持ってい

ます。

1. 出力から数えて奇数番目の NOR ゲートは、AND ゲートと考えられる。ただし変数は反転する。
2. 出力から数えて偶数番目の NOR ゲートは OR ゲートと考えられる。

NAND の場合と同じ様に、次のことが言えます。同様に証明するために、最初のゲートに入る入力を A , B とします。

一段目の出力　二段目のあとのの NOR 出力は $\overline{A+B}=\bar{A}\bar{B}$ となります。

二段目の出力　次の NOR の入力として、上で求めた式と新しい入力変数として C を考えますと、その出力として $\overline{\bar{A}\bar{B}+C}=(A+B)\bar{C}$ が得られます。ここで A , B に注目しますと、OR になっています。これは出力から数えて偶数番目の入力は出力で OR となっていることになっており、証明が出来ました。

三段目の出力　三段目の出力はもう一つの未知の入力を D とします。そうすると三段目の出力は $\overline{(A+B)\bar{C}+D}=(\bar{A}\bar{B}+C)\bar{D}$ となります。

以上の結果から、先程の二つの規則が出てきます。
このことについての例を、図 6.3 に示します。

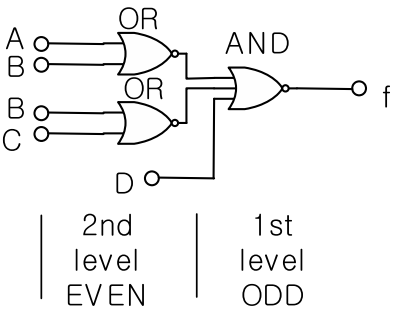


図 6.3　NOR 回路の例：$(A+B)(\bar{B}+C)\bar{D}$

6.2 組み合わせ回路の合成

回路の合成は、次のような手順で行われます。

1. 問題の意味を正確に捉え、その意味を代数式で表現する。
2. 代数式を簡略化する。
3. 代数式を論理回路で作る。

これらの手順を例題で示していくことにします。

NAND 回路による合成例

$f = AB + D(B + \bar{C})$ を合成してみます。NAND 回路の性質を利用しますと、出力から奇数番目は OR 回路ですので AB と $D(B + \bar{C})$ とが最終段の入力となります。最終段から 2 番目は AND 回路ですので、AB の方は 1 つの NAND 回路で実現できます。もう一方は D と $(B + \bar{C})$ とを入力としても NAND 回路で実現できます。次が B と $\bar{C}$ ですが、これらの変数は奇数番目の入力ですので、反転しなければなりません。以上から次の回路図 6.4 が得られることになります。

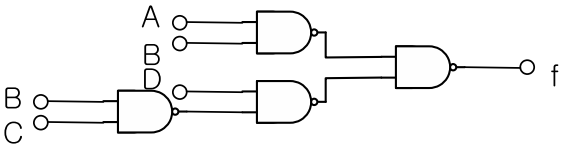


図 6.4 NAND 回路による合成例：$f = AB + D(B + \bar{C})$

NOR 回路による合成例

$f = (\bar{A}B + A)(\bar{A}B + \bar{B})$ を合成してみます。NOR 回路の性質を利用しますと、出力から奇数番目は AND 回路ですので $(A\bar{B} + A)$ と $(\bar{A}B + \bar{B})$ とが最終段の入力となります。最終段から二番目は OR 回路ですので、$(\bar{A}B + A)$ の方は、$\bar{A}B$ と A の入力を持つ NOR 回路で実現できます。もう一方は、$\bar{A}B$ と $\bar{B}$ とを入力としてもつ NOR 回路で実現できます。ここで $\bar{A}B$ は共通であること、更にこの式はもう一つの NOR で実現できることと、これらの変数は奇数番目の入力であるので、反転しなければならなりません。以上から次の回路図 6.5 が得られることになります。

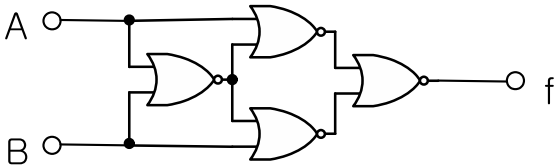


図 6.5　NOR 回路による合成例：$f = (\bar{A}B + A)(\bar{A}B + \bar{B})$

第 7 章

順序論理回路

順序論理回路は、その回路の中に記憶装置を持っており、そのために出力はその時点の入力ばかりでなく、過去の入力の情報も出力に影響を与えます。

ここでは主に組み合わせ論理回路を用いた非同期式論理回路を簡単に説明します。実際の論理回路で多用されている同期式順序回路については、この後の章で述べることにします。

7.1 解析

実際の論理回路においては、信号は瞬時に伝わることはなく、ある遅れ時間を持って出力に現れます。この値は数ナノ秒という小さい時間ではありますが、その数ナノ秒の時間であっても考慮しなければならない場合があります。特に高速に変化する信号の場合には、非常に重要な影響を与えることになります。

次の図 7.1 に示す回路を考えてみます。

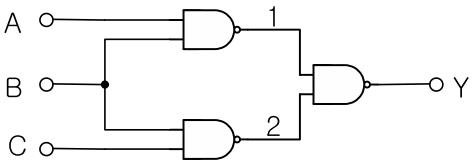


図 7.1　遅延を考えるための回路例

この回路において各々の NAND 回路が 20 [*nsec*] の遅れを持っているとしますと、回路全体の遅れは 40 [*nsec*] となります。この遅れは各々の NAND 回路において生じるのですが、計算を簡単にするために全体の遅れ Δt がまとめて出力側に現れるとします。これを図 7.2 に示しておきます。

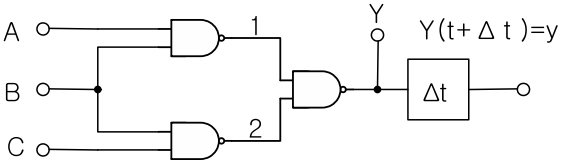


図 7.2　遅延を表現した等価回路

この図において *Y* は遅延がないと仮定した場合の出力、*y* は遅延が存在する場合の出力ということになります。このように遅延がない場合と遅延が存在する場合とに分けて考えることによって、回路の解析は易しくなります。

次に図 7.3 のように出力が入力側へ戻る場合について考えてみます。

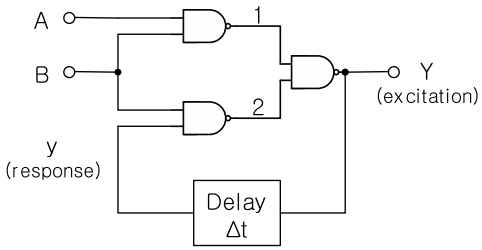


図 7.3 帰還がある回路

この場合も全ての遅れ時間は、一カ所に集中していると仮定しています。実際にはあり得ないことではありますが、帰還回路の解析の説明を簡単にするためにこのような仮定をしています。このとき次のような式が成立しなければなりません。

$$Y = AB + By \tag{7.1}$$

$$y(t) = Y(t + \Delta t) \tag{7.2}$$

図 7.3 の場合のタイミング図を図 7.4 に示しておきます。この図を見て分かりますように、Y の出力に対して Δt の遅れで y の出力が現れます。このタイミング図をカルノー図で表現しますと、図 7.5 のように得られます。

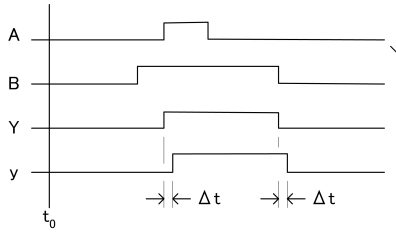


図 7.4 図 7.3 のタイミング図

A / y \ B	00	01	11	10
0	0	0	1	0
1	0	1	1	0

Y

図 7.5 図 7.3 のカルノー図

このカルノー図の中に y が含まれていることに注意する必要があります。

このようなカルノー図のことを励振マトリックスと呼んでいます。励振マトリックスの下に Y という記号が記載されているのは、このマトリックスが Y の出力を表していることを示しています。つまり Y の状態が回路を励振する直前であることを意味しています。この励振マトリックスは、(7.1) 式を表現しており (7.2) 式は含まれていません。(7.2) 式は、Y の値が時間的にずれるだけですので Y の値が y の値と等しい場合には、その後回路は同じ状態を保ち安定状態となりますが、Y の値が y の値と異なると回路は元の状態にとどまらず、別の状態へと変化することになりますので、不安定状態となってしまいます。

A , B は、制御入力と呼ばれ、回路の外部からの信号によって決められます。励振マトリックスは、(7.1) 式で示す帰還のない場合の入出力関係を示しているだけですが、帰還がある場合の変化についてもこの励振マトリックス上に動きを記入することによって表現することが出来ます。なぜなら励振マトリックスは、帰還がないときの回路の状態を全て示しているからです。

このことからこの励振マトリックスの制御入力の変化は、入力 y の変化は起きませんので、励振マトリックス上において水平方向への移動に対応しています。それに対して Y の値の変化は、入力 y の変化を起こしますので、励振マトリックスの垂直方向の移動に対応していることになります。

励振マトリックスだけでは、回路がどの様に変化しているかを 知ることは出来ないことは以上の説明から分かります。回路の動的な動きをカルノー図上に記述したのが、次の表 7.1 で示す転移マトリックスです。

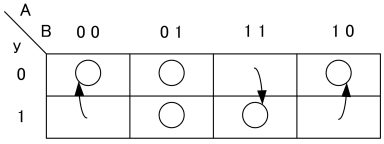


表 7.1　転移マトリックス

転移マトリックスの矢印が入っているところを見ますと、矢印の根本の状

態を (7.1) 式へ代入すると Y の値がそのときの状態から変化してしまうことが分かります。Y の値が変化するということは、この回路の場合には、y の値が変化することを意味していますので、矢印の頭の方へ変化しなければなりません。このようにして転移マトリックスは構成されています。つまり転移マトリックスは、ある状態になったとき、変化が起こる場合の転移先を示していることになります。

表 7.2 は、フローマトリックスと呼ばれています。このフローマトリックスは、これまで述べてきた内容をカルノー図上に記載した表となっています。マトリックス上の数値は、回路の状態を区別するための数字です。よって同じ数字は、同じ回路状態を示しています。また数字を丸で囲んであるのは、その状態になったとき、異なった制御入力が入らない限り同じ状態を保つ安定した状態であることを示しています。

A / B / y	00	01	11	10
0	①	②	4	③
1	1	⑤	④	3

表 7.2 フローマトリックス

いずれの図も制御入力の動的な変化については、記載されていないことに注意する必要があります。それは制御入力は、外部の未知な信号によって決まる内容ですので、このような図で表現することは不可能であることによります。

以上のことから、解析はフローマトリックスの完成によって 終了となります。この図を見ることによって与えられた回路動作の全ての情報が得られることになるからです。

7.2　合成

合成の問題は、先にも述べたように様々な結論が得られるのですが、回路合成の一つの手順としては、回路解析の手順を逆にたどることによって得られます。

全く逆ではありませんが、回路合成の手順を述べますと、次のようになります。

1. 要求される内容からフローテーブルと呼ばれる表を求める
2. この表の最小化を行う
3. 転移図を求める
4. フローマトリックスを求める
5. 励振マトリックスを求める
6. ブール代数式を求める
7. 回路図を描く

この手順を説明するために、次のような問題について考えていくことにします。

例
入力が 0 , 1 , 0 , 1 と変化するとき 2 進数で数え上げ、2 ビットで出力を表示する（入力が変化する度に数える）回路を順序回路で求めて下さい。ただし最初の状態は 0 であるとします。

合成の手順に従って、回路の合成を行っていきます。例の内容に従って、まず始めにフローテーブルを作成します。2 ビットであるので出力は、Z_1 , Z_2 の 2 変数が必要になります。最初の状態は、$Z_1 = 0$, $Z_2 = 0$ です。またこの例の場合入力は一つ、出力は二つですので、次のフローテーブルが得られます。

	入力 A		出 力	
	0	1	Z_1	Z_2
a	①	2	0	0
b	3	②	0	1
c	③	4	1	1
d	1	④	1	0

表 7.3　例のフローテーブル

この表 7.3 の中で一番左の記号は、状態を区別するために付けた記号です。また丸印を付けた数字と、何も付いていない数字は、丸印のある方が安定状態、何も付いていない数字は不安定な状態であることを示しています。よって 1 番から始まり、入力に 1 が入りますと 2 番へと変化するのですが、次の 0 が入って 3 番へと変化しますが、この時は入力に 1 が入ったわけではありませんので、出力はカウントしてはなりません。次に入力に 1 が入ると 4 番目に移動しますが、そこは不安定なので、安定なところへと移動します。更に入力が 0 となると、元の状態へと戻ります。他埋まり入力も出力も全て 0 の状態に戻ることになります。

この例で難しいところは、0 入力であっても出力の状態が変化することです。それを避けるためには本体の出力とは異なる新しい変数 X , Y を導入する必要があります。

これらの変化は、次の図 7.6 に示す転移図によって明確になります。

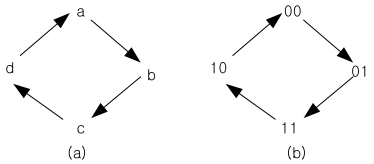


図 7.6　例の転移図

以上の結果からフローマトリックスを求めていくのですが、今までの結果から分かりますように入力は一つですが、出力は二つ必要です。この出力のうち入力が入る前の状態を各々 x , y としますとフローマトリックスは、表 7.4 のように得られます。この図は A , x , y の各々に対応する状態に対する状態の番号を代入することによって得られます。

フローマトリックスから励振マトリックス を求めなければなりませんが、これは次のように行うことによって得られます。まず始めに図 7.6 の（ｂ）において各状態に対して 2 進数を当てはめていますが、この 2 進数を当てはめることを 2 次割り当てと呼んでいます。この場合にはたまたま表 7.3 の出力と同じような数値となっていますので注意する必要があります。

表 7.4 のフローマトリックスの各状態に相当する 2 次割り当てを転移図 7.6 上の 2 次割り当てを選んでいきます。このようにして表 7.5 の励振マトリックスが得られます。最後に励振マトリックスを用いて各セルの中の左側の数字が X、右側の数字が Y を表していますので、この励振マトリックスから次のようなブール代数式が得られることになります。

$$X = \bar{A}y + Ax \tag{7.3}$$

$$Y = \bar{A}y + A\bar{x} \tag{7.4}$$

x y \ A	0	1
(a) 00	①	2
(b) 01	3	②
(c) 11	③	4
(d) 10	1	④

表 7.4　例のフローマトリックス

x y \ A	0	1
00	0 0	0 1
01	1 1	0 1
11	1 1	1 0
10	0 0	1 0

表 7.5　例の励振マトリックス

以上の結果から図 7.7 に示す回路図が得られます。

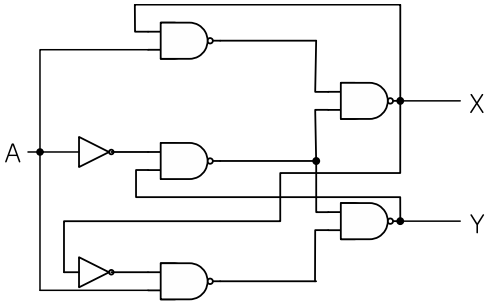


図 7.7　(7.3)(7.4) 式の回路図

この結果で終了したわけではなく、まだ出力 Z_1 , Z_2 が表現されていませんので、その表現を求める作業が残っています。これはフローマトリックス表 7.4 とフローテーブル表 7.3 を用いて、$A=0$, $x=y=0$ のとき $Z_1=0$, $Z_2=0$ となる安定な値を取ります。$A=0$, $x=0$, $y=1$ のときは、不安定なので $Z_1=1$, $Z_2=1$ となる安定な値を取ります。$A=0$, $x=1$, $y=1$ のときは、安定なので $Z_1=1$, $Z_2=1$ となる安定な値を取ります。この様な手順を繰り返して表 7.6 のように求められます。

x \ A / y	0	1
00	0 0	0 1
01	1 0	0 1
11	1 0	1 1
10	0 0	1 1

表 7.6　出力の励振マトリックス

この場合のブール代数は、次のように与えられます。

$$Z_1 = \bar{A}y + Ax = X \tag{7.5}$$
$$Z_2 = A \tag{7.6}$$

(7.3) 式〜(7.6) 式を用いますと、図 7.8 の回路図が得られます。

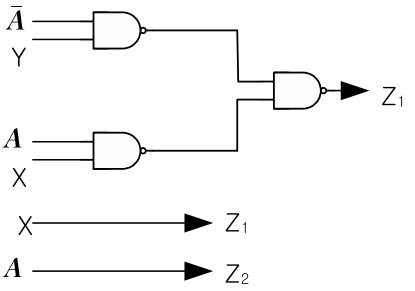


図 7.8　(7.5)(7.6) 式の回路図

求める回路図は、図 7.7 と図 7.8 とを合成した回路図によって得られます。このような手順にて、求める回路図が得られますが、実現される回路図は 2 次割り当てを様々に変化されせることによって、もっと別な形の回路図も得ることが出来ることを忘れてはいけません。

第 8 章

同期式順序回路

第 7 章で取り扱った順序回路は、組み合わせ論理回路のみを用いた回路の解析及び合成でした。よって実際の問題に対しては、現在あまり用いられることはありませんが、実現したのは非同期回路であり、その手法は非同期回路を取り扱う場合において重要となります。

ここではフリップフロップを用いた順序回路の解析及び合成の問題について考えます。この章においても始め解析について説明し、次に合成の問題を考えていくことにします。

8.1　解析

解析を進めるに当たって、例を示しながら説明していくことにします。その例とは図 8.1 に示す回路です。

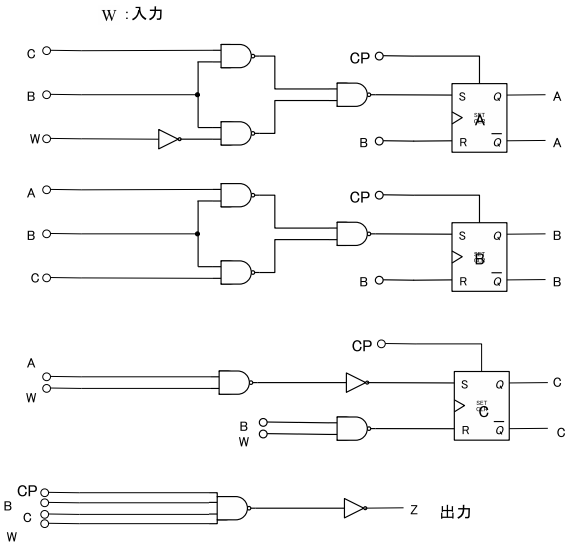


図 8.1　解析のための回路例

このような回路の解析を進める方法として、状態図 を用いる方法がありますのでそのことから説明します。状態図とはクロックパルスが入ったとき、どの様な状態へ変化するかということを図示したものであり、この回路図の場合の状態図は、図 8.2 のように与えられます。

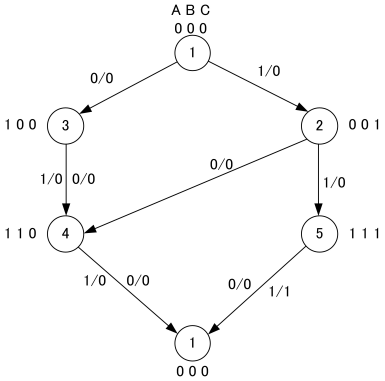


図 8.2 回路例に対する状態図

この状態図を求めるためには、まず始めに回路図から代数式を求める必要があります。図 8.1 から次のように求められます。この式の中で、左辺はフリップフロップ S , R の状態を示しています。また添え字は各々フリップフロップ A , B , C を示しています。このように左辺にフリップフロップの状態を持ってくることによって、後の解析がやりやすくなります。

この式の中には、フリップフロップ自体の動作を記述した式は、含まれていないことに注意する必要があります。またフリップフロップの出力のところが A , B , C となっていることも注意しなければなりません。つまりこれらの式とフリップフロップの式とにより A , B , C の値が決定されるということです。

$$
\begin{aligned}
S_A &= \bar{B}C + \bar{B}\bar{W} \\
R_A &= B \\
S_B &= A\bar{B} + \bar{B}C \\
R_B &= B \\
S_C &= \bar{A}W \\
R_C &= B + \bar{W} \\
Z &= BCW\,(CP)
\end{aligned}
$$

状態図は、通常 0 の状態から始めます。状態図の線上に書かれた数値は、左上が入力 W を示し、右下が出力 Z を示しています。

クロック CP が入ってこなければこの回路は、そのときの状態を保っています。もし変化するとすれば回路自体で発振している異常事態が発生していることになります。

図 8.2 において A , B , C が 0 , 0 , 0 の状態から出発するとして、クロックが入ってきますと $W = 0$ の場合と $W = 1$ の場合がまずは発生します。この二つの入力の場合 $A = B = C = 0$ ですので、上で求めた式から次の表 8.1 が得られます。

	$A = B = C = 0$ and $W = 0$	$A = B = C = 0$ and $W = 1$
S_A	1	0
R_A	0	0
S_B	0	0
R_B	0	0
S_C	0	1
R_C	1	0
Z	0	0

表 8.1　例題の状態表

この表とここで用いているＳＲフリップフロップの特性表から、$W = 0$ の場合には、A , B , C が 1 , 0 , 0 の状態となり $W = 1$ の場合には、A , B , C が 0 , 0 , 1 の状態となります。この結果から状態図上の 3 と 2 とが得られることになります。

このような手順を繰り返していくことによって状態図が完成されますが、3 のようにこの点からは一つの枝しか出ていません。これはいずれの場合も同じ状態へと変化するため、一つにまとめて状態を示しているためです。また状態 2 から 4 へと線が引かれているのは、同じ状態を表現するのに別の点を取る無駄を省くためです。更に最後は元の 0 , 0 , 0 の状態に戻っています。状態の数には限りがありますので、いずれの場合でもいずれかは元の状

態に戻ります。

この場合にはＳＲフリップフロップを用いているにもかかわらず、ＳＲフリップがもっている不定な状態にはいることはありませんでした。しかし場合によってはそのような状態に入り込むことも考えられます。そのようにならないように解析は十分に行うべきです。例えば A , B , C が 0 , 1 , 0 の状態から出発すると、このような不定の状態に入り込みます。各自調べてみて下さい。

8.2 合成

ある命題が与えられたとき、その命題を実現するための回路合成の問題について考えます。

次のような例を考えます。

> 3 ビットパルス列の中に、1 が奇数個あるとき出力を出す回路を設計します。ただし 3 ビット続いた後は、初期状態に戻るとします。

まず始めにフリップフロップがいくつ必要であるかということを考えなければなりません。フリップフロップは一つで二つの状態を取ることが出来ますので、この場合のように三つのパルスを用いて 8 個の状態を表現するためには、三つのフリップフロップが必要になります。

一般に m 個の状態を表すために必要なフリップフロップの数 n には、次の関係式が必要です。

$$2^n > m \tag{8.1}$$

回路を合成するためには、状態図を作ることから始めなければなりませんが、通常 0 , 0 , 0 の状態から始めます。この選択は一般的に電子機器は、動作させるために電源を投入することから始まり、その状態は通常 0 から始まると考えられるため、妥当な選択であるとは思われますが、電子機器の中には、そのように動かない品物も存在するので、一概には言えないことです。

状態図を描く場合には、回路の中がどの様になっているかということを知らなくても描くことが出来ます。なぜならば状態図はある一つの状態に対して入力が入って来たとき、その結果さえ分かれば描けるからであり、回路合成の問題というのは、入力と出力の関係から始まるのが普通だからです。

よって回路の具体的な状態に関わらず、まず始めに 1 番から出発し、入力に対して結果が現れるいう状態の変化に応じて 2 番 3 番・・・というように番号を振っていくことにします。入力は 0 もしくは 1 がランダムに出現しますので、一つの状態から枝が二つ発生することになります。 3 段目以降枝が一つしか出ていないのは、すでに可能性はないためです。また 3 ビット後には初期状態に戻ることになっていますので、図 8.3 の状態図が得られます。

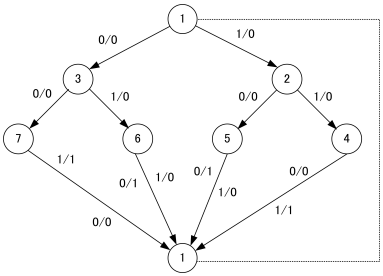


図 8.3　例の状態図

このようにして状態図が得られたのですが、実はこの状態図 8.3 は冗長です。このことは次のような状態表 8.2 を作ることによって分かります。

	次の状態		出力	
	入力		入力	
現在の状態	0	1	0	1
1	3	2	0	0
2	5	4	0	0
3	7	6	0	0
4	1	1	0	1
5	1	1	1	0
6	1	1	1	0
7	1	1	0	1

表 8.2　例題の状態表

この表 8.2 は、次のようにして作られています。一番左側は、単に状態の番号を表しています。次の二つの列は、入力が 0 の場合と 1 の場合の転移先の状態の番号を示しています。一番右の 2 列は、入力が 0 の場合と 1 の場合の出力 Z の状態を示しています。

このようにして表を作りますと、現在の状態の番号で言って 4 番と 7 番、5 番と 6 番とは同じ入出力状態の番号が並んでいることが分かります。よって 6 番と 7 番は省略することが出来、このようにして省略した状態でもう一度状態図を描きますと図 8.4 が得られます。

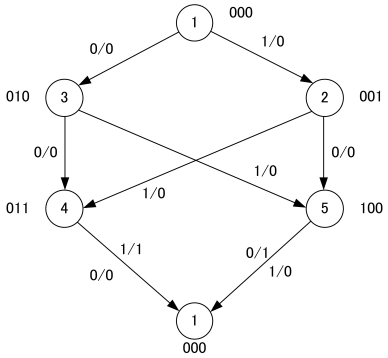


図 8.4　最終の状態図

この次に行う作業は、いわゆる 2 次割り当てという作業です。これは先ほども出てきましたが、各状態に対して回路図のどの状態を当てはめるかと言うことです。動作の始まりには、通常 0 の状態を当てはめます。

この例の場合五つの状態が必要であるので、(8.1) 式から 3 個のフリップフロップが必要になります。この例において仮に次のような表 8.3 を決めます。

	現在の状態			次の状態	
	フリップフロップ			入力	
	A	B	C	0	1
1	0	0	0	010	001
2	0	0	1	100	011
3	0	1	0	011	100
4	0	1	1	000	000
5	1	0	0	000	000
-	1	0	1	-	-
-	1	1	0	-	-
-	1	1	1	-	-

表 8.3 例題の 2 次割り当て表

この次に行うべきことは、どのフリップフロップを用いるかということですが、ここではＪＫフリップフロップを用いることにします。

上で求めた 2 次割り当て表とＪＫフリップフロップの励振表を用いて、制御マトリックスを作らなければなりません。この制御マトリックスはフリップフロップが三つあるので、三つの制御マトリックスが必要になってきます。フリップフロップ A の場合について考えてみます。2 次割り当て表とＪＫマトリックスの励振表とから、A , B , C の現在の各状態から入力 W が 0 の場合と 1 の場合とに応じて求める必要があります。例えば現在の状態が $A=1$, $B=0$, $C=0$ のとき、$W=0$ の場合 $A=1$ から $A=0$ へと転移します。これはＪＫマトリックスの励振表では $J=\phi$, $K=1$ となり、$W=1$ では $A=1$ は $A=0$ となりますので、$J=\phi$, $K=1$ となり同じ結果となります。

別の例で現在の状態が $A=0$, $B=1$, $C=0$ のとき $W=0$ の場合 $A=0$ から変化しません。これはＪＫマトリックスの励振マトリックスでは $J=0$, $K=\phi$ となり、$W=1$ では $A=0$ から $A=1$ へと転移します。これはＪＫフリップフロップの励振マトリックスから $J=1$, $K=\phi$ となります。

Q^n	$\rightarrow$	Q^{n+1}	J	K
0		0	0	ϕ
0		1	1	ϕ
1		0	ϕ	1
1		1	ϕ	0

表 8.4　JK フリップフロップ励振表

このような手順を繰り返していくと、次のような三つの表 8.5、表 8.6、表 8.7 が得られることになります。

C W \ A B	00	01	11	10
00	0 ϕ	0 ϕ	−	ϕ 1
01	0 ϕ	1 ϕ	−	ϕ 1
11	0 ϕ	0 ϕ	−	−
10	1 ϕ	0 ϕ	−	−

表 8.5　$J_A - K_A$

C W \ A B	00	01	11	10
00	1 ϕ	Φ 0	−	0 ϕ
01	0 ϕ	ϕ 1	−	0 ϕ
11	1 ϕ	ϕ 1	−	−
10	0 ϕ	ϕ 1	−	−

表 8.6　$J_B - K_B$

C W \ A B	00	01	11	10
00	0 ϕ	1 ϕ	−	0 ϕ
01	1 ϕ	0 ϕ	−	0 ϕ
11	ϕ 0	ϕ 1	−	−
10	ϕ 1	ϕ 1	−	−

表 8.7　$J_C - K_C$

次に行わねばならない作業は、表 8.5、表 8.6、表 8.7 を用いて各々のフリップフロップに対する制御マトリックスを完成することです。

フリップフロップは、３個あるので８通りの状態を実現することが出来ますが、必要な状態の数は５個です。よって３個の状態が使われないで残っています。これらの３個の状態に対して W が２個の状態を取りますので、制御マトリックスの中に６個の空のセルが存在することになります。合計６個の状態が余ることになりますが、この場合には、このセルはどの様な状態をとっても問題が生じませんので、冗長なセルと考えることが出来ます。そこでこのセルに ϕ を記入して、もう一度書くと表 8.8、表 8.9、表 8.10 が得られます。

C W \ A B	00	01	11	10
00	0 ϕ	0 ϕ	ϕ ϕ	ϕ 1
01	0 ϕ	1 ϕ	ϕ ϕ	ϕ 1
11	0 ϕ	0 ϕ	ϕ ϕ	ϕ ϕ
10	1 ϕ	0 ϕ	ϕ ϕ	ϕ ϕ

表 8.8 $J_A - K_A$ の制御マトリックス

C W \ A B	00	01	11	10
00	1 ϕ	Φ 0	ϕ ϕ	0 ϕ
01	0 ϕ	ϕ 1	ϕ ϕ	0 ϕ
11	1 ϕ	ϕ 1	ϕ ϕ	ϕ ϕ
10	0 ϕ	ϕ 1	ϕ ϕ	ϕ ϕ

表 8.9 $J_B - K_B$ の制御マトリックス

C W \ A B	00	01	11	10
00	0 ϕ	1 ϕ	ϕ ϕ	0 ϕ
01	1 ϕ	0 ϕ	ϕ ϕ	0 ϕ
11	ϕ 0	ϕ 1	ϕ ϕ	ϕ ϕ
10	ϕ 1	ϕ 1	ϕ ϕ	ϕ ϕ

表 8.10 $J_C - K_C$ の制御マトリックス

これらの表に応じて、次のブール代数式が得られます。

$$J_A = \bar{B}C\bar{W} + B\bar{C}W \tag{8.2}$$
$$K_A = A \tag{8.3}$$
$$J_B = \bar{A}\bar{C}\bar{W} + CW \tag{8.4}$$
$$K_B = C + W \tag{8.5}$$
$$J_C = B\bar{W} + \bar{A}\bar{B}W \tag{8.6}$$
$$K_C = B + \bar{W} \tag{8.7}$$

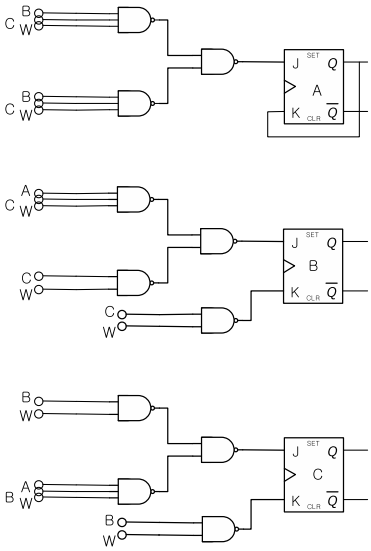


図 8.5　例題の回路網

これらの方程式が導かれると、この式を用いて回路図 8.5 を描くことが出来ますが、これだけで回路合成が終わったわけではありません。すなわち出力 Z に関する回路が抜けているのです。この出力に関するマトリックスは、図 8.4 から次表 8.11 のように求められます。

C W \ A B	00	01	11	10
00	0	0	ϕ	1
01	0	0	ϕ	0
11	0	1	ϕ	ϕ
10	0	0	ϕ	ϕ

表 8.11　出力の制御マトリックス

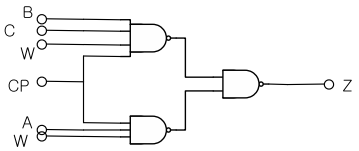


図 8.6　出力回路

出力の制御マトリックスから、出力に対する方程式は、次のように求めることが出来ます。

$$Z = A\bar{W} + BCW \tag{8.8}$$

この式を用いて出力回路は、図 8.6 のように求めることが出来ます。

図 8.5 と図 8.6 とをあわせることによって、求めようとしていた回路全体が完成することになります。

以上の合成手順を纏めると、次のようになります。

1. 命題を明確にします。
2. 状態図を求めます。
3. 状態表を求め簡略化します。
4. 簡略化された状態図を求めます。
5. ２次割り当てを行います。

6. フリップフロップを選びます。
7. 各フリップフロップごとに制御マトリックスを求めます。
8. 制御マトリックスの空白を埋めます。
9. ブール代数を求めます。
10. 回路図を求めます。
11. 出力の制御マトリックスを求めます。
12. 出力のブール代数を求めます。
13. 出力の回路図を求めます。
14. 二つの回路図を一緒にし、完成させます。

第 9 章

加減算・掛け算

ここではデジタル回路の基本的な回路である加減算回路および掛け算器について、解説することにします。

9.1　加算回路

二つの数を加える場合、加えあわせることによって桁があがる場合と、桁上げが起こらない場合とが存在します。このような事実を回路で実現するには、桁上げが起こる場合と起こらない場合とを分けて考える必要があります。桁上げが起こるとしても、加算しようとしている数値にその下の桁からの桁上げが存在している場合と、下の桁からの桁上げが存在しない場合の２種類が考えられます。下の桁からの桁上げが存在する場合には、入力変数として加算しようとしている数値プラス桁上げの項も考えなければなりませんので、結局入力変数としては、三つの変数が必要になります。それに対して下の桁からの桁上げがない場合には、変数としては二つになります。

下の桁からの桁上げが起こらない回路はハーフアダー と呼ばれており、下の桁からの桁上げが発生するような、より一般的な加算回路をフルアダーと呼んで区別しています。

まず始めに簡単なハーフアダーの説明を行い、次に一般的なフルアダーに

ついて説明を行うことにします。

9.1.1　ハーフアダー

ハーフアダーは、説明したように下の桁からの桁上げが発生しない場合ですので、その計算過程の真理値表を求めますと、入力を A_i , B_i、出力を C_{i+1} , S_{i+1} とすると表 9.1 のようになります。ここで C_{i+1} は桁上げの数値、S_{i+1} は加算後の同じ桁での値です。

入力		出力	
B_i	A_i	S_{i+1}	C_{i+1}
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

表 9.1　ハーフアダー

この真理値表が出来れば、これから方程式を求めることは簡単で、次の式のように求めることが出来ます。S_{i+1} の場合も C_{i+1} の場合も 1 となるところを求めることになります。

$$S_{i+1} = A_I\overline{B_i} + \overline{A_i}B_i = A_i \oplus B_i \tag{9.1}$$

$$C_{i+1} = A_iB_i \tag{9.2}$$

この式を用いて NAND 回路を用いて回路を構成すると、図 9.1 のように求められます。

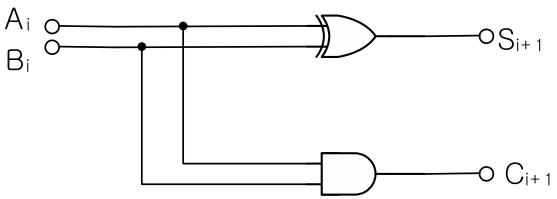


図 9.1 XOR を用いたハーフアダー

ハーフアダーの回路が求まりましたので、次はフルアダーについて述べていきます。

9.1.2 フルアダー

次に下の桁から桁上げが存在する場合の一般的な加算器について考えることにします。このフルアダーは、実はハーフアダーを組み合わせることによって構成することが可能であることが分かります。

入力変数は 3 個になりますが、真理値表としては、表 9.2 のようになります。

入力			出力	
C_i	B_i	A_i	S_{i+1}	C_{i+1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

表 9.2 フルアダー

真理値表だけを見ていたのでは、複雑すぎてよく分かりませんが、式で書くと次のように与えられます。まず S_i について求めると

$$\begin{aligned} S_i &= \overline{C_i}\,\overline{B_i}A_i + \overline{C_i}B_i\overline{A_i} + C_i\overline{B_i}\,\overline{A_i} + C_iB_iA_i \\ &= \overline{C_i}(\overline{B_i}A_i + B_i\overline{A_i}) + C_i(\overline{B_i}\,\overline{A_i} + B_iA_i) \\ &= \overline{C_i}(\overline{B_i}A_i + B_i\overline{A_i}) + C_i\overline{\overline{B_i}A_i + B_i\overline{A_i}} \\ &= C_i \oplus (B_i \oplus A_i) \end{aligned} \tag{9.3}$$

次に C_i について求めますと

$$\begin{aligned} C_i &= \overline{C_i}B_iA_i + C_i\overline{B_i}A_i + C_iB_i\overline{A_i} + C_iB_iA_i \\ &= (\overline{C_i} + C_i)B_iA_i + C_i(\overline{B_i}A_i + B_I\overline{A_i}) \\ &= B_iA_i + C_i(A_i \oplus B_i) \\ &= \overline{B_iA_i\,\overline{(S_iC_i)}} \end{aligned} \tag{9.4}$$

このように変形することが出来ますので、今までの結果とあわせますと、図 9.2 のように二つのハーフアダーを用いることによってフルアダーが得られることになります。

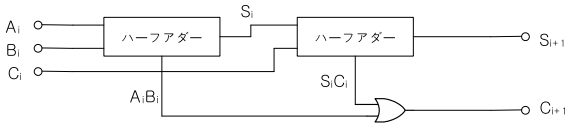


図 9.2　フルアダーの回路

9.1.3　多数桁の加算

多くの桁の加算を行うためには、二つの方法があります。一つは 1 個のフルアダーを用いる方法と、もう一つは多数のフルアダーを用いる方法です。

直列加算

1 個のフルアダーを用いる手法で、直列シフトレジスタに蓄えられている[*1]二つの数値を LSB（Least Significant Bit の頭文字で、日本語では最下位ビットの意味）から順にフルアダーに送り、その出力をアキュムレータへと送り込み、加算の結果を得ます。この回路のブロック図は、図 9.3 として与えられます。

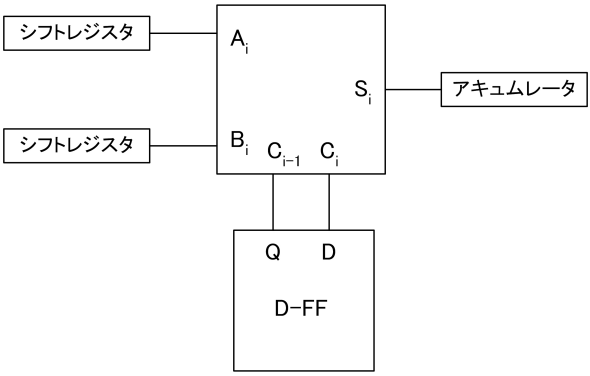


図 9.3 直列加算

図から分かりますように回路全体を制御する必要があります。つまり加算されるシフトレジスタの動きとフルアダー、桁上げするために必要なＤフリップフロップそれとアキュムレータとは同期を取って制御する必要があります。そのために別に制御回路が必要となってきます。また 1 桁ごとに加算が行われることによってこの後の説明で出てくる並列加算に較べると加算速度が遅くなります。

[*1] シフトレジスタについては 10 を参照下さい。

並列加算

多数の必要な数だけフルアダーを用いて加算を行う手法を、並列加算と呼んでいます。４ビットの並列加算器を図 9.4 に示します。図の中で C_3 はオーバーフローを示しています。図を見ると分かりますように、 加算は並列で処理されますので、直列加算に較べて格段に処理速度は速くなります。

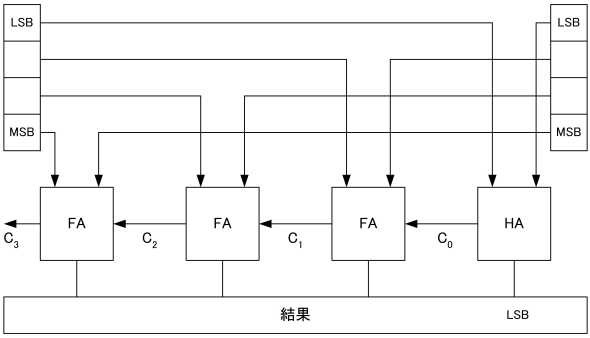


図 9.4　４ビット並列加算器

9.2　減算回路

減算の場合も加算と同じように、半減算（ハーフサブトラクター)、全減算（フルサブトラクター）を考える必要があります。まず始めにハーフサブトラクター、次にフルサブトラクターを考えていきます。

9.2.1　ハーフサブトラクター

ハーフサブトラクターは、ハーフアダーと同様に上の桁からの桁下げが起こらない場合と言うより、上の桁を考えない場合の演算から求められま

す。その演算規則は次のように与えられます。[*2]

入力		出力	
被減数	減数	差	借り
A_i	B_i	D_i	E_i
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

表 9.3 ハーフサブトラクター

この真理値表から、次の式が得られます。

$$D_{i+1} = A_i \oplus B_i \tag{9.5}$$

$$E_{i+1} = \overline{A_i} B_i \tag{9.6}$$

この式を用いるとハーフサブトラクターは、図 9.5 として得られます。この図と図 9.1 とを比較しますとハーフアダーの AND ゲートの入力にインバータが一つ入っていることだけが違っていることが分かります。

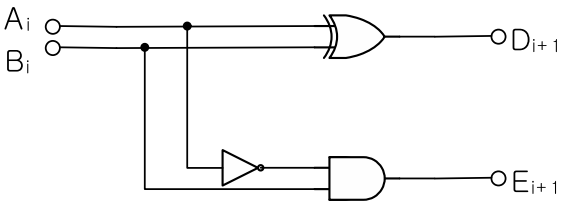


図 9.5 ハーフサブトラクター

[*2] この表の中で右から 2 行目は、差になっていることに注意

9.2.2　フルサブトラクター

フルサブトラクターは、借りる入力も考慮した真理値表から得ることが出来ます。その真理値表は、次のように与えられます。

入力			出力	
A_i	B_i	E_i	D_{i+1}	E_{i+1}
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

表 9.4　フルサブトラクター

この真理値表から、回路図はハーフサブトラクターを用いて、図 9.6 のように求められます。

この図 9.6 と図 9.2 とを比較しますと、ハーフアダーがハーフサブトラクターになっているだけで、全く同じ構成となっていることが分かります。

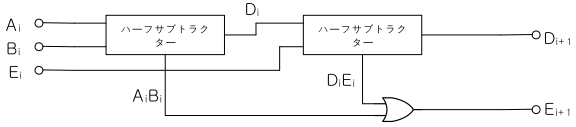


図 9.6　フルサブトラクター

9.2.3 並列減算器

並列減算器について説明します。並列減算器は、並列加算器と全く同じ回路構成で実現することが出来ます。もちろんハーフアダーの代わりにハーフサブトラクター、フルアダーの代わりにフルサブトラクターを用いる必要があります。

４ビットの場合の並列減算器を、図 9.7 に示しておきます。

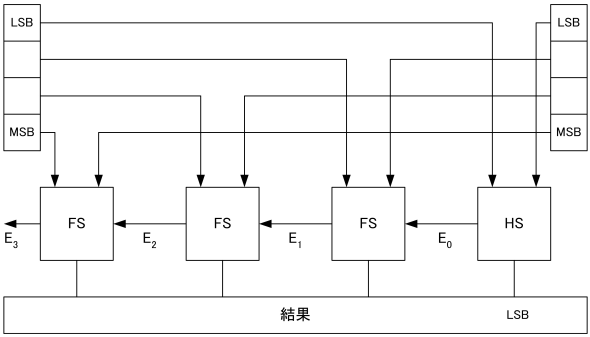


図 9.7 並列減算器

9.2.4 加算器を用いた減算

加算器を用いることによっても、減算を行わせることが可能です。現実問題として、加算器・減算器が入り交じっているよりも、加算器のみで回路が構成されている方がすっきりして望ましいことです。

加算器による減算とは２進数の特徴を利用した手法です。1101 から 0100 を引くという例を考えてみます。結果は 1001 となりますが、引く数である 0100 の補数は 1011 となります。この数と 1101 とを加えますと 11000 となります。この得られた数値の一番上の桁の数値を一番下の桁に加えると

1001 となって、求める結果が得られたことになります。

このことを図で示すと、図 9.8 のように与えられます。

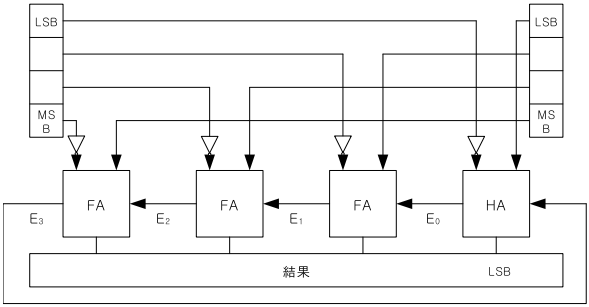


図 9.8　加算器を用いた減算

9.3　乗算回路

デジタル信号処理では、乗算を直接行うことは出来ません。その代わり乗算は加算の繰り返しによって行えることを利用して乗算が行われます。ここでは加算とシフト法、繰り返し加算を取り上げ説明していきます。

9.3.1　加算とシフト法

２進法での乗算は、１０進法と同様にして、次のように行うことが出来ました。この例では４ビットの乗算を考えています。

$$
\begin{array}{rr}
 & 1101 \\
\times & 1010 \\
\hline
 & 0000 \\
 & 11010 \\
 & 000000 \\
 & 1101000 \\
\hline
 & 10000010
\end{array}
$$

この例から分かることは、ゼロを掛ける場合の結果は全てゼロの値となり、掛けるゼロの値の桁のところまで桁上げしていると言うことです。それと掛ける数値が 1 の場合の結果だけが加算されることになります。

これらの事実から回路は加算器と、桁上げに必要な加算のための制御装置とが必要となります。この回路は次の図 9.9 の回路によって与えられることが分かります。

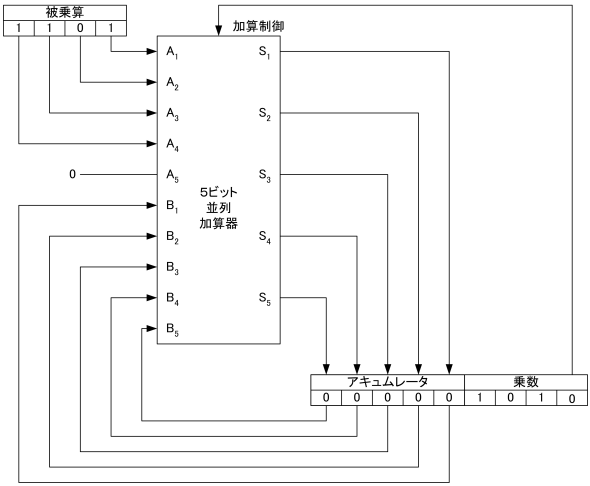


図 9.9　加算とシフト法

図 9.9 の回路動作は、次のようになります。

1. 被乗数は、左上のレジスタに蓄えられ、乗数は右下のレジスタに蓄えられています。
2. 始めにアキュムレータレジスタは、ゼロになっているとあひます。乗数のＬＳＢが、５ビット加算器の加算制御入力に印可されますが、値がゼロであるので、加算は実行されません。よってアキュムレータレジスタはゼロのままとなります。

3. 次にアキュムレータレジスタと乗数レジスタとが一つ右にシフトされます。この状態では、乗数レジスタのＬＳＢは 1 となりますので、加算制御には 1 が印可され加算動作が行われます。このときアキュムレータ 00000 に被乗数レジスタからの値 01101 が加算されます。
4. 次にアキュムレータレジスタと乗数レジスタとが右に一つだけシフトしますとアキュムレータレジスターの左端にはゼロが入ってきます。このとき乗数レジスタのＬＳＢはゼロですので、加算は行われません。
5. 更にアキュムレータレジスターと乗数レジスタとが右に一つシフトしますと、今度は乗数レジスタのＬＳＢが 1 となりますので、アキュムレータレジスタに被乗数レジスタの値が加算されることになります。
6. アキュムレータレジスタと乗数レジスタとが右に一つシフトしますと、今度は乗数レジスタのＬＳＢがゼロですので加算は行われません。
7. 結果はアキュムレータレジスタと乗数レジスタとにまたがって得られ、010000010 と得られます。

9.3.2　くりかえし加算

この計算方法は $6 \times 5 = 6+6+6+6+6$ であることを利用した手法です。つまりこの場合の例で言うと、6 という数値を繰り返し 5 回加えることによって解答を求める手法と言うことになります。このことから図 9.10 のようなブロック図によって掛け算を行うことが出来ます。

この回路ブロックは、それほど説明をしなくても理解できると思われますので省略します。

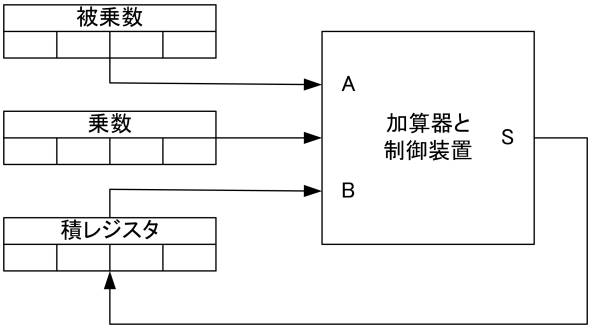


図 9.10　くりかえし加算

第 10 章

カウンタ

カウンタは、クロックパルスとして入ってきた信号の数をカウントする装置です。カウンタとしては多数の種類が考案されており、２進カウンタ、２進化１０進カウンタ（１０進カウンタあるいはＢＣＤカウンタとも呼ばれる)、ランダムカウンタ、同期カウンタ、リップルカウンタ、リングカウンタ、シフトレジスタなどがあります。ｎビットのカウンタとは、2^n 個のパルスが数えられるカウンタのことを言っています。よってｎビットのカウンタは、2^n 個の状態が必要になります。

10.1　同期式２進カウンタ

同期式２進カウンタとは、フリップフロップに同時期にパルスが入力されて、同時期に動作するため、その名前が付けられています。例えば４ビットのカウンタを考えてみますと、１６個のパルスを数えることが出来ることになりますが、そのためには１６通りの異なった状態が必要になります。

この場合の状態図は、簡単に描くことが出来て、図 10.1 のように与えられます。

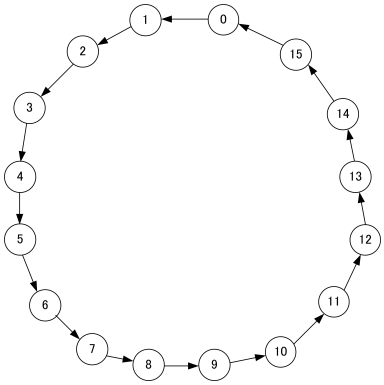


図 10.1　4 ビットカウンタの状態図

この状態図を回路に実現するのですが、この手順は順序回路の合成で用いた手法を使うことになります。まず始めに状態図 10.1 を用いて、状態表の作成を行います。この状態表は、次に示す表 10.1 として与えられます。

現在の状態					次の状態				
状態	A	B	C	D	状態	A	B	C	D
0	0	0	0	0	1	0	0	0	1
1	0	0	0	1	2	0	0	1	0
2	0	0	1	0	3	0	0	1	1
3	0	0	1	1	4	0	1	0	0
4	0	1	0	0	5	0	1	0	1
5	0	1	0	1	6	0	1	1	0
6	0	1	1	0	7	0	1	1	1
7	0	1	1	1	8	0	0	0	0
8	1	0	0	0	9	1	0	0	1
9	1	0	0	1	10	1	0	1	0
10	1	0	1	0	11	1	0	1	1
11	1	0	1	1	12	1	1	0	0
12	1	1	0	0	13	1	1	0	1
13	1	1	0	1	14	1	1	1	0
14	1	1	1	0	15	1	1	1	1
15	1	1	1	1	0	0	0	0	0

表 10.1　４ビットカウンタの状態表

この例の場合、各状態に対して A , B , C , D の値が既に割り当てられていますので、改めて２次割り当てを行う必要はなくこれらの値をそのまま用いることが出来ます。

この状態表から、４ビットであるのでフリップフロップは４個必要になります。その４個のフリップフロップの制御マトリックスは、ＪＫフリップフロップを用いて、表 10.2 から表 10.5 に示すように得られます。

C\AB / D	0 0	0 1	1 1	1 0
0 0	**0 ϕ**	**0 ϕ**	**ϕ 0**	**ϕ 0**
0 1	**0 ϕ**	**0 ϕ**	**ϕ 0**	**ϕ 0**
1 1	**0 ϕ**	**1 ϕ**	**ϕ 1**	**ϕ 0**
1 0	**0 ϕ**	**0 ϕ**	**ϕ 0**	**ϕ 0**

表 10.2　$J_A - K_A$ 制御マトリックス

C\AB / D	0 0	0 1	1 1	1 0
0 0	**0 ϕ**	**ϕ 0**	**ϕ 0**	**0 ϕ**
0 1	**0 ϕ**	**ϕ 0**	**ϕ 0**	**0 ϕ**
1 1	**1 ϕ**	**ϕ 1**	**ϕ 1**	**1 ϕ**
1 0	**0 ϕ**	**ϕ 0**	**ϕ 0**	**0 ϕ**

表 10.3　$J_B - K_B$ 制御マトリックス

C\AB / D	0 0	0 1	1 1	1 0
0 0	**0 ϕ**	**0 ϕ**	**0 ϕ**	**0 ϕ**
0 1	**1 ϕ**	**1 ϕ**	**1 ϕ**	**1 ϕ**
1 1	**ϕ 1**	**ϕ 1**	**ϕ 1**	**ϕ 1**
1 0	**ϕ 0**	**ϕ 0**	**ϕ 0**	**ϕ 0**

表 10.4　$J_C - K_C$ 制御マトリックス

C\AB / D	0 0	0 1	1 1	1 0
0 0	**1 ϕ**	**1 ϕ**	**1 ϕ**	**1 ϕ**
0 1	**ϕ 1**	**ϕ 1**	**ϕ 1**	**ϕ 1**
1 1	**ϕ 1**	**ϕ 1**	**ϕ 1**	**ϕ 1**
1 0	**1 ϕ**	**1 ϕ**	**1 ϕ**	**1 ϕ**

表 10.5　$J_D - K_D$ 制御マトリックス

表 10.2 から表 10.5 から、次の式が得られる。

$$J_A = K_A = BCD \tag{10.1}$$
$$K_B = K_B = CD \tag{10.2}$$
$$J_C = K_C = D \tag{10.3}$$
$$J_D = K_D = 1 \tag{10.4}$$

この式から 4 ビット同期式カウンタの回路図は、次の図 10.2 のように与えられます。この図から分かりますようにさらに多くのビット数を持つカウンターは、ＪＫフィリップフロップを追加することによって構成することが出来ます。

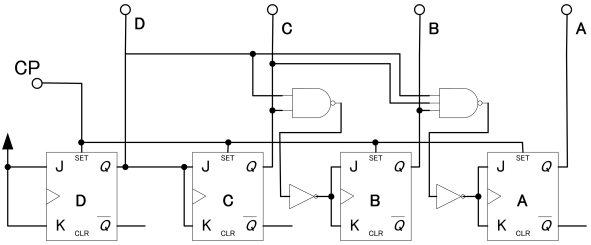


図 10.2　４ビット同期式カウンタ

10.2　リップルキャリイの同期式２進カウンタ

動作周波数を気にしなければ、直感的にカウンタは回路合成の手法を取らなくてもフリップフロップの特性表から簡単に構成することが出来ます。ＪＫフリップフロップの場合、ＪＫ端子を共に V_{cc} あるいはＧＮＤとすることにより、クロックが入ってくる度に出力が反転する状態になったり、変化しなかったりしますので、そのことを利用してカウンタを構成することが出来ます。４ビットの場合の同期カウンタを、図 10.3 に示します。

同期式カウンタと呼ばれるのは、クロックが全てのフリップフロックを同時に起動しているためです。信号は左から順に伝わっていくので、動作周波数としては小さくならざるを得ません。逆にその分だけ NAND ゲートのファンイン数を小さくすることが出来ます

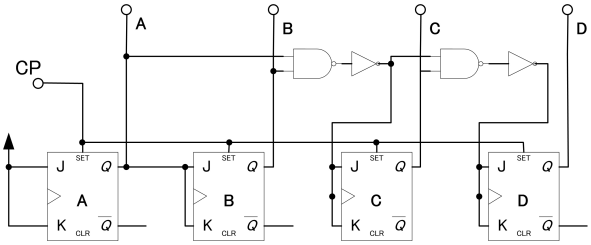


図 10.3　リップルキャリイの４ビット同期カウンタ

10.3　リップルカウンタ

このカウンタは、同期式ではなく非常に簡単な回路構成で実現することが出来ます。このカウンタもＪＫフリップフロップの特性表から得られますが、出力を次のフリップフロップのクロックとして用いています。

多数ビットのカウンタは、同じ回路をつないでいくことによって簡単に得られます。４ビットの場合について、リップルカウンタの回路図を図 10.4 に示しておきます。

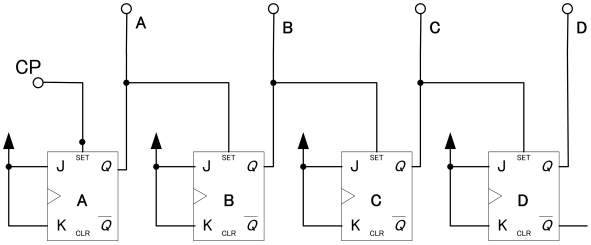


図 10.4　４ビットのリップルカウンタ

10.4　２進化１０進カウンタ

このカウンタは、１０進数に等価な２進数を割り当てるようにカウントされるカウンタのことを指しており、その代表である 8-4-2-1 ＢＣＤ同期カウンタは、それぞれの重みを考慮して１０進数に直すことが出来ます。通常のカウンタと違うところは、値が 1111 となった後再び 0000 に戻る必要があるということです。回路合成手法としては、４ビットのカウンタと全く同じですので、ここではこの 8-4-2-1 ＢＣＤ同期カウンタの回路を図 10.5 に、リップルキャリイを用いてフリップフロップＤに入力される入力数を減らしたカウンタとしてリップルキャリイ型ＢＣＤカウンタを、図 10.6 に示しておきます。

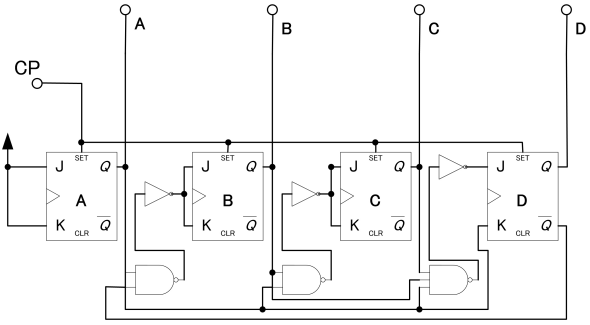


図 10.5　8-4-2-1 ＢＣＤ同期カウンタ

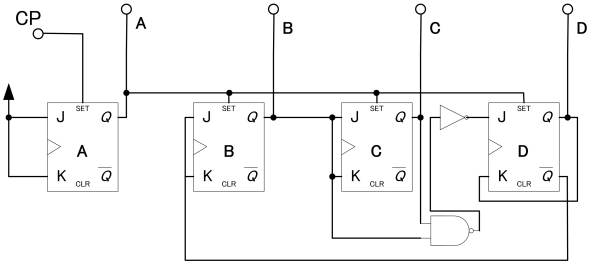


図 10.6　8-4-2-1 ＢＣＤカウンタ（リップルキャリイ型）

10.5　シフトレジスタ

シフトレジスタとは、クロックが入るたびにデータがずれていく回路です。この意味は言葉で説明するよりも次の表 10.6 タイミングチャートを見てもらうことによってはっきりと分かると思います。この場合入力データとして、1 , 1 , 0 , 0 が入ってきた場合について考えています。

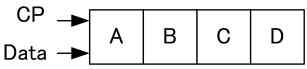


図 10.7　４ビットシフトレジスタ

		FF の状態			
CP	Data	A	B	C	D
		0	0	0	0
1	1	1	0	0	0
2	1	1	1	0	0
3	0	0	1	1	0
4	0	0	0	1	1

表 10.6　４ビットシフトレジスタ

このシフトレジスタ回路の実現は、非常に簡単です。もし A が 1 である時にクロックが入ってきますと B は 1 になり、A が 0 である時にクロックが入ってくると B は 0 になりさえすればよいので、A に対して B がどの様に変化するかという制御マトリックスさえあればよいことになります。シフトレジスタの残りの部分は、この制御マトリックスを繰り返すことになるからです。ＪＫフリップフロップを用いた制御マトリックスは、表 10.7 のように与えられます。

B \ A	0	1
0	0　ϕ	1　ϕ
1	1　ϕ	ϕ　0

表 10.7　制御マトリックス

この表によるとＪＫフリップフロップノ入力として、次の式を満足すればよいことが分かります。

$$J = A$$
$$K = \bar{A}$$

つまりＪＫフリップフロップノ入力としては、J 側は前段の値そのもの、K 側は前段入力の反転であればよいことになります。このことからシフトレジスタの一番最初は、入力そもののとそれを反転させた入力とが必要になりますが、それから先はフリップフロップ自体が同時に反転出力も出力するため、単に従属に接続していけばよいことになります。

ここで４ビットのシフトレジスタを、図 10.8 に示しておきます。

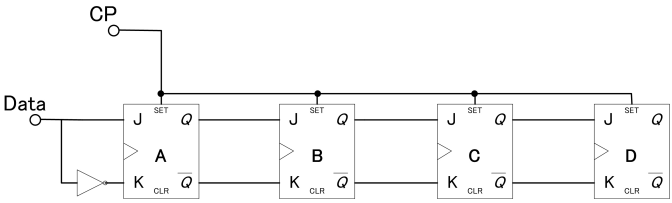


図 10.8　4 ビットのシフトレジスタ

10.6　分周回路

周波数を何分の 1 とかに落としたい場合があります。このような回路を分周回路と呼んでいます。簡単に想像できるように、分周回路は 2 進カウンタを用いることにより実現することが出来ます。分周回路の例を図 10.9 に示します。またこの分周回路のタイミング図を図 10.10 に示しておきます。

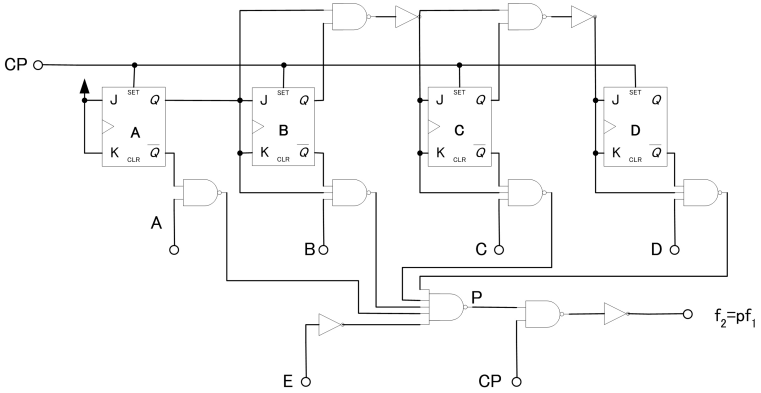


図 10.9　4 ビット分周回路

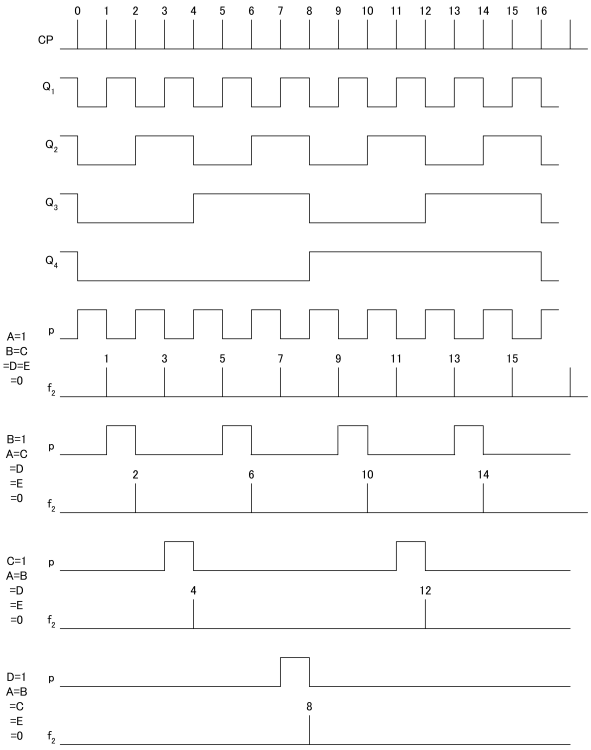


図 10.10　タイミング図

この回路図を用いると、実は任意の分周を得ることが出来ます。クロックパルス（ＣＰ）の周波数を f_1 とします。また出力の周波数を f_2 としますと、分周係数 p を用いて、次の関係式となります。

$$f_2 = pf_1 \qquad 0 \leq p \leq 1 \tag{10.5}$$

ここで入力 A , B , C , D , E を係数 p に等しい２進数に選ぶことによって、任意の分周回路が得られます。

入力 $A = 1$ でその他の入力が全てゼロである場合を考えてみます。このとき出力としては、タイミングチャートからも分かりますように

1 , 3 , 5 , 7 , 9 , 11 , 13 , 15 の奇数番目のパルスのみが現れます。これは $p = 2^{-1}$ であることを示しています。

入力 $B = 1$ でその他の入力がゼロである場合を考えますと、2、6、10、14、18、22、26 番目のパルスが得られます。これは $p = 2^{-2}$ であることを示しています。

入力 $C = 1$ でその他の入力がゼロである場合には、4 , 12 , 20 , 28 , 36 番目のパルスが得られ、これは $p = 2^{-3}$ です。

また $D = 1$ の場合は $p = 2^{-4}$ となり、$E = 1$ の場合は $p = 2^{-5}$ となります。

これらの結果は、タイミングチャートを見るとパルスが重なっていないことから NAND を考えますと、これらのパルスが全て加えあわされた結果が得られることになります。例えば $A = B = 1$ で、その他はゼロの場合には、$p = 2^{-1} + 2^{-2} = 0.75_{(10)}$ のパルス出力が得られます。また $B = C = 1$ で、その他の入力がゼロの場合には、$p = 2^{-2} + 2^{-3} = 0.373_{(10)}$ が得られます。

$0.75_{(10)} = 0.11_{(2)}$, $0.373_{(10)} = 0.011_{(2)}$ ですので、$0.ABCDE$ と並べることによって、入力を 2 進数として並べた数値と分周係数とが一致していることが分かります。

例

入力周波数の $0.15625_{(10)}$ 分周係数を持つ回路の入力を求めてみます。

まず最初に、この数値を 2 進数へ変換する必要があります。その結果として $0.00101_{(2)}$ が得られます。この結果から $C = E = 1$ その他の入力をゼロと置くことによって、目的の分周回路を得ることが出来ます。

この分周回路を実現するために、図 10.9 をそのまま用いても良いのですが、フリップフロップは、信号を伝送するために必要ですけれども 1 ， 2 ，4 番目の 0 から 1 への転移は行われないので不必要となり、回路は少し簡単な図 10.11 によって実現することが出来ます。

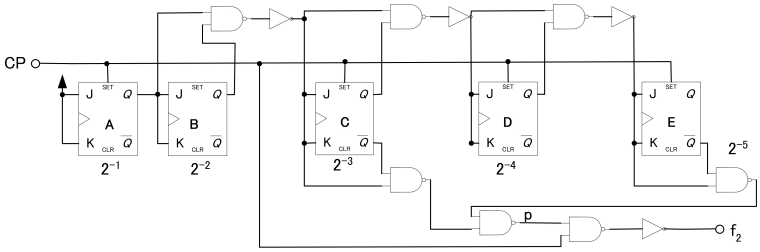


図 10.11　分周係数 $0.15625_{(10)}$ の回路

第 II 部

応用編

第 11 章

集積回路

実際のデジタル回路がどの様な形で構成され、販売されているかということについて、具体的な論理回路で示していくことにします。

論理回路は、大きく四つにタイプに分類することが出来ます。その四つとは、DCTL(Diode -Coupled Transistor Logic)、DTL(Diode Transistor Logic)、TTL(Transistor to Transistor Logic)、CML(Current mode Logic) です。世の中にはこれ以外の名称が多数用いられていますが、それらの名称は、色々な会社が自社向けとして付けた名称で、上記の名称の変形回路になります。

ここで取り上げている回路は、かなり古い回路ですが、デジタル回路を学ぶことには問題がありません。

11.1　基本論理回路

ここでは四つの基本論理回路の代表的な回路を上げておきます。

11.1.1　DCTL

この回路の基本構成は、次のような形式の回路です。

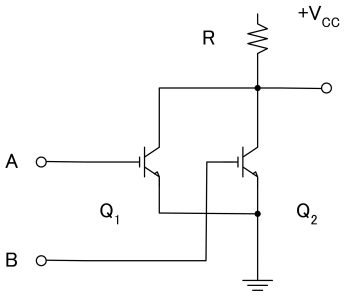


図 11.1　ＤＣＴＬ

この回路は、入力の電圧と出力の電圧を真理値表で表現しますと、次のようになりますので NOR 回路を実現することになります。

入力 A	入力 B	出力
0	0	1
0	1	0
1	0	0
1	1	0

表 11.1　NOR

11.1.2　DTL

この回路の基本構成は、次のような形式の回路です。

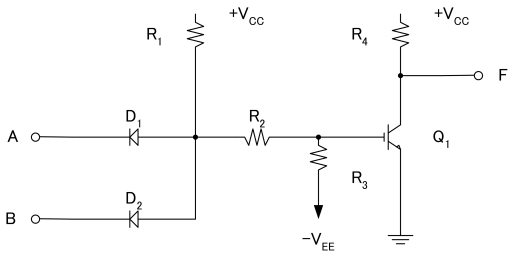


図 11.2 ＤＴＬ

この回路は、入力の電圧と出力の電圧を真理値表で表現しますと、次のようになりますので NAND 回路を実現することになります。

入力 A	入力 B	出力
0	0	1
0	1	1
1	0	1
1	1	0

表 11.2 DTL

11.1.3 TTL

この回路の基本構成は、次のような形式の回路です。

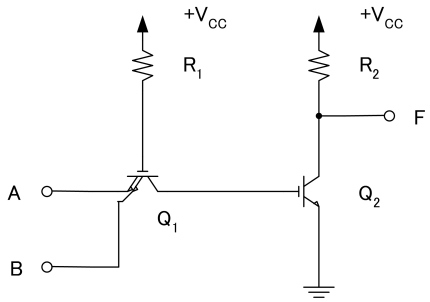


図 11.3　ＴＴＬ

この回路は、入力の電圧と出力の電圧を真理値表で表現しますと、次のようになりますので NAND 回路を実現することになります。

入力 A	入力 B	出力
0	0	1
0	1	1
1	0	1
1	1	0

表 11.3　TTL

11.1.4　CML

この回路の基本構成は、次のような形式の回路です。

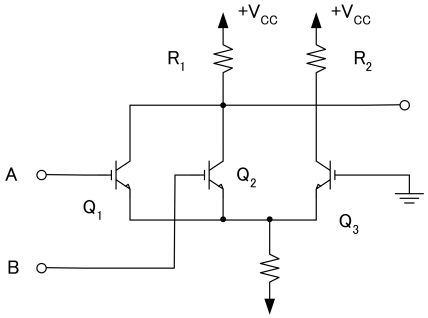


図 11.4　ＣＭＬ

この回路は、入力の電圧と出力の電圧を真理値表で表現しますと、次のようになりますので NOR 回路を実現することになります。

入力 A	入力 B	出力
0	0	1
0	1	0
1	0	0
1	1	0

表 11.4　CML

11.2　各社の論理回路

ここでは、論理回路の最初の段階ではありますが、各社の代表的な論理回路について紹介していきます。

11.2.1　Fairchild 社

Fairchild 社は、1961 年図 11.1 に示す DCTL 回路の各トランジスタのベースに抵抗を付加した図 11.5 に示す IC を発表しました。この回路は、RTL(Resistor Transistor Logic) として有名な回路です。

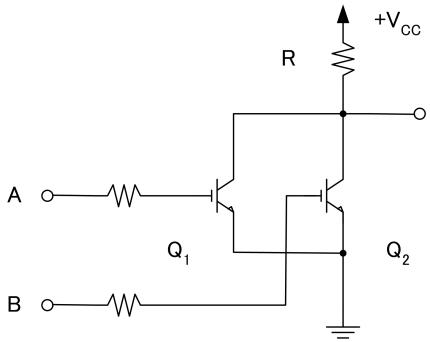


図 11.5　RTL 回路

このベース抵抗は、過電流を防ぐために挿入されています。回路が簡単であり消費電流が小さくて済むことから、人気となった回路です。

1964 年 DTμL シリーズとして、次に示す回路が発表されました。

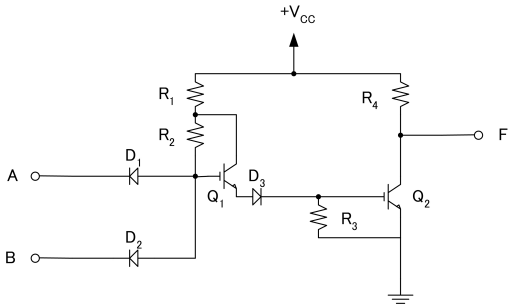


図 11.6　DTμL 回路

この回路は、Signetics 社の回路 11.9 に較べて消費電力や応答速度において優れています。これは入力側のトランジスタが飽和しないように設計されているために、常に活性領域として動作することに依存しています。またそのために余計な電流が流れることもないため、消費電力も少なくなっています。ただし出力がコレクタ出力となっていますので、ファンアウト数をそれほど高くすることは出来ません。また当然外部の影響を受けやすいという欠点もあります。

11.2.2　Texas Instruments 社

1961 年 Texas Instruments 社によって発表された RCTL (Resistor Capacitor Transistor Logic) を、次の図に示します。

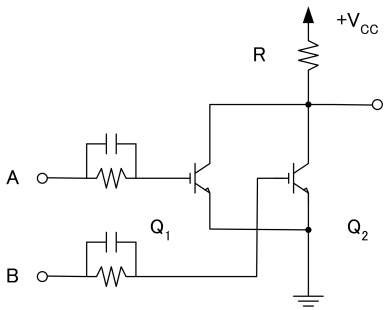


図 11.7　RCTL 回路

この回路は、ベース側の抵抗に並列の形で容量が挿入されていることが特徴です。この容量によってベース側のインピーダンスが高周波領域で低下するため若干応答速度が速くなります。並列となっている抵抗の値は、容量の影響分高めに設定することが可能となりますので、消費電流を低く押さえることが可能となりますが、逆に低くなりすぎると f_T が低下して来ますのでその分速度が遅くなってしまいます。

この IC は、当時としては初めて 10 ピンのフラット・パッケージに入れられ、SN51 シリーズとして販売されました。

1963 年 Texas Intruments 社は、改良型 DTL 論理回路 SN53 シリーズを発表しました。その回路を、次の図に示します。

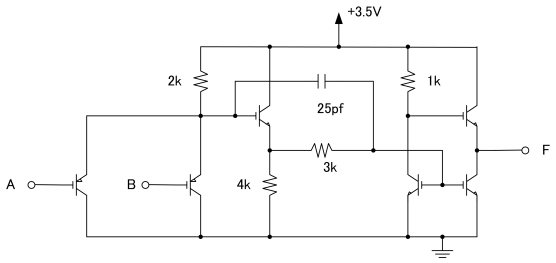


図 11.8　改良型 DTL

この IC は、入力側が PNP トランジスタのベースとなっていますので、入力インピーダンスが大きく、しかも利得を持っています。容量が入っていますので、エミッタフロアとなっているトランジスタが温情対からオフ状態へと変化するとき最終段のカレントミラーへのバイアス電圧が強制的に急激に負となりますのでオフ状態への変化が急速に行われます。しかしこの回路によって余計な消費電流が必要となってしまいます。

出力がエミッタ出力となっていますので、インピーダンスが低くなります。その結果ドライバ能力が向上し、ファンアウト数を高く取ることが出来るようになります。またインピーダンスが低くなることにより他からの影響が少なくなります。

この IC は、集積回路の特徴を巧く利用して多数のゲートを一つのチップ上で実現したため、J-K フリップフロップを実現した最初の IC として知られています。

11.2.3　Signetics 社

Signetics 社が 1960 年始めに発表しました画期的な論理回路が、図 11.9 です。この回路は簡単な回路構成となっているにも関わらず、図 11.5 図 11.7 に較べて回路設計が容易で大振幅動作が可能でありしかも雑音に強いといった特徴を持っています。

ただしコレクタ側の抵抗値が大きいため、深く飽和に入ってしまい応答速度が遅くなるという欠点があります。

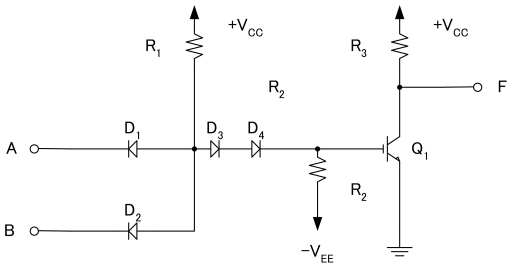


図 11.9　Signetics 社の DTL

Sygnetics 社は、次に示すような回路も発表しました。

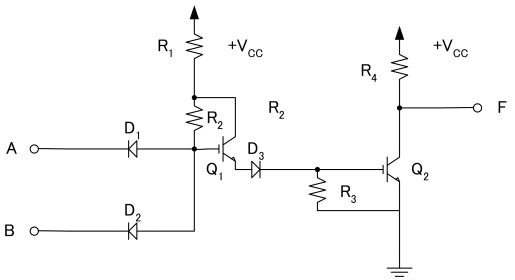


図 11.10　改良型 DTL 回路

図から分かりますように、ダイオードの一部をトランジスタで置き換えた構造となっています。こうすることによって出力トランジスタのベース抵抗が低下し、その結果出力トランジスタが飽和したときのインピーダンスを下げることが出来ます。そのためファンアウト数が向上します。

11.2.4　Motorola 社

Motorola 社は、図 11.11 に示す回路を発表しました。この回路は、MECL(Motorola emitter-coupled logic) と呼ばれています。

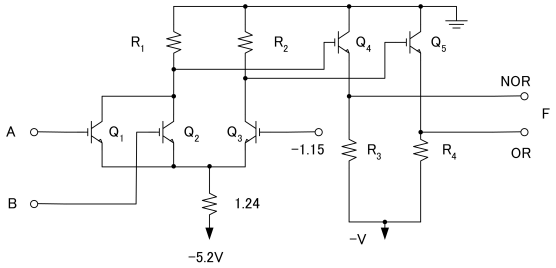


図 11.11　MECL 回路

この回路形式は、非飽和型の ECL 回路であるため非常に高速に動作します。出力がエミッタフォロアであるため、非常に大きなファンアウト数を持つことが可能となります。ただし入力がベース入力のためインピーダンスが高く雑音を拾いやすい構造となっています。

1965 年に Motorola 社は、VTL(Variable Threshold Logic) と呼んでいる回路を発表しました。この回路を、次の図に示しておきます。

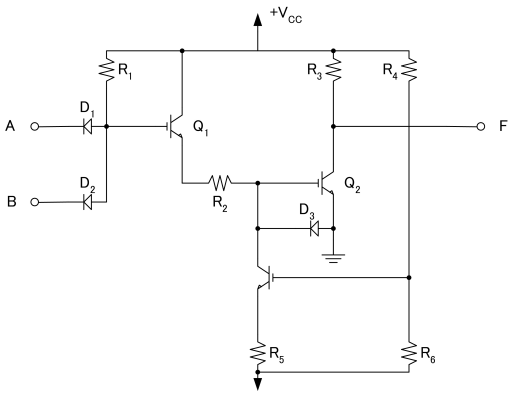


図 11.12　VTL 回路

図から分かりますように、電源電圧を変化させますと、下側にあるトランジスタの電流が変化します。その結果出力段のトランジスタが動作する状態が変化しますので、スレッショルド電圧を変化することが出来ます。 このことを利用して、スレッショルド電圧が異なる回路を接続することが可能となります。

11.2.5　Sylvania 社

1963 年 Sylvania は、次の図に示す T^2L(Transistor-to-transistor logic) を発表しました。

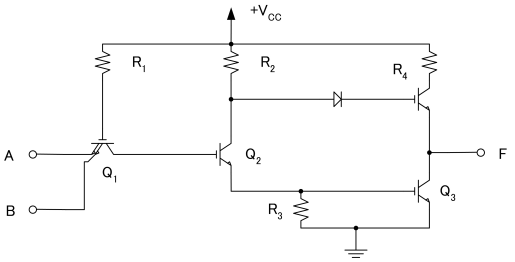


図 11.13　T^2L 回路

この回路は、図を見ると分かりますように IC ならではの回路となっています。この様にダブル・エミッタのような回路を最初に用いた会社が Sylvania 社です。入力側がベース接地となっていますので、非常に高速に動作させることが出来ます。また出力側のエミッタ出力は、オンあるいはオフの状態でも上側のトランジスタは生きていますので、常に低インピーダンスとなっています。そのため外部の影響を受けることが少なくなります。

第 12 章

メモリ

近年デジタルＩＣの発展はめざましいものがあります。この理由は、デジタルＩＣとしての利点によるところが大です。 このデジタル化発展の中で、メモリの役割は非常に大きなものがあります。ここでは、このメモリについて述べることにします。

12.1　種類

メモリの種類を考える場合においても、様々な分類方法が存在します。これらを考えてみると、次のような分類が存在します。

信号入出力　RAM(Random Access Memory)　信号を任意の番地に書き込み、読み出しすることが出来る。

ROM(Read Only Memory)　信号の読み出しだけが出来る。

記憶の仕方　**揮発性**　電源を切ると信号がクリアされ、消えてしまうメモリ

不揮発性　電源を切っても信号は残って消えないメモリ

材料　MOS メモリ　MOS トランジスタを用いたメモリ

Bipolar メモリ　Bipolar トランジスタを用いたメモリ

磁気メモリ　磁化の方向により情報を記録する

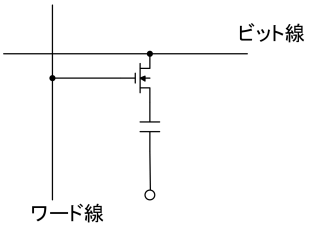


図 12.3　DRAM

12.2　各種メモリ

スタティック RAM(SRAM)　フリップフロップを用いたメモリで、 次の図 12.1 図 12.2 に示す 2 種類があります。

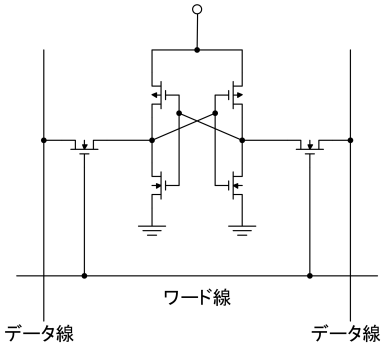


図 12.1　6Tr-SRAM

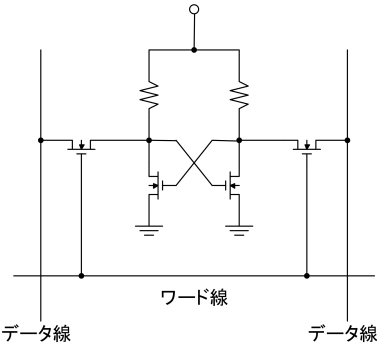


図 12.2　4Tr-2R-SRAM

ダイナミック RAM(D-RAM)　MOSFET と容量とを一つずつ用いたメモリ。図 12.3

ROM(Read Only Memory)　読み出すことは出来るが、書き込むことが出来ないメモリ

マスク ROM　メモリを製造する工程において用いられるガラスマスク上に焼き付けて作成するメモリ

EPROM(UV-Erasable PROM)　書き込みを行い、書き込んだデータを紫外線を用い消去することが可能な ROM

EEPROM(Electrically Erasable PROM)　電気的に消去することの出来るメモリ

EAROM(Electrically Alterable ROM)　電気的に消去することの出来るメモリ

FEEPROM(Flash Electrical Erasable PROM)　電気的に消去することの出来るメモリ

その他　シンクロナス DRAM　インターフェイスの部分も同期を取って高速にしたメモリ

ラムバス DRAM　ランバス社が開発したメモリ、インターフェイスを 改良して高速にしている

フィールド/フレームメモリ　TV や VTR などの画像処理用途のメモリ

ビデオ RAM/マルチポート DRAM　コンピュータ画像バッファ用のメモリ

12.3　メモリの動作

64 [bit] のメモリ動作について考えてみます。まず始めにこのメモリの概要について図 12.4 に示します。

アドレス	ビットD	ビットC	ビットB	ビットA	アドレス	ビットD	ビットC	ビットB	ビットA
ワード 0					ワード 8				
ワード 1					ワード 9				
ワード 2	1	0	1	1	ワード 10				
ワード 3					ワード 11				
ワード 4					ワード 12				
ワード 5					ワード 13				
ワード 6					ワード 14				
ワード 7					ワード 15				

図 12.4　メモリの構成

図 12.4 の中で空白になっているところが、メモリ素子を示しています。この場合４ビットから構成される１６個のグループとなっており、この内４ビットを一纏めにしてワードと呼びます。 この様な構成となっているメモリを 16×4 メモリ構成と呼んでいます。ワードの場所のことをデータのアドレスと呼んでいます。この様な構成は、ROM の場合も RAM の場合も同じです。

16×4 メモリ構成である 7489RAM の外部回路を、次の図 12.5 に示しておきます。

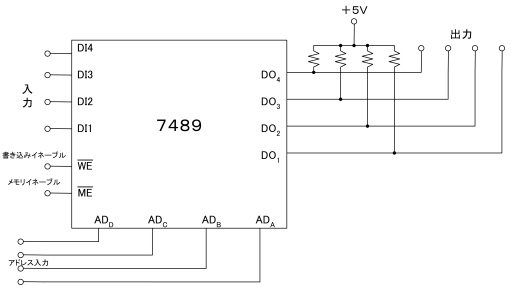


図 12.5　7489 ブロック図

7489RAM の端子の状態を現す表は、表 12.1 のようになります。

動作モード	入力	
	$\overline{WE}$	$\overline{ME}$
書き込み	0	0
読み出し	1	0
保持	ϕ	1

表 12.1　7489 の動作表

図 12.4 のようにワード 2 の場所に 1011 を書き込む場合、データ入力は、$DI_4 = 1$、$DI_3 = 0$、$DI_2 = 1$、$DI_1 = 1$、アドレス入力は、$AD_D = 0$、$AD_C = 0$、$AD_B = 1$、$AD_A = 0$、書き込みイネーブルは $\overline{WE} = 0$ として、最後に メモリイネーブルを $\overline{ME} = 0$ とすることにより、アドレス 2 の場所、第 2 ワードに 1011 が書き込まれます。

読み出しの場合、$\overline{WE} = 1$、$\overline{ME} = 0$ とすれば、出力として $\overline{DO_1} = 0$、$\overline{DO_2} = 1$、$\overline{DO_3} = 0$、$\overline{DO_4} = 0$ が得られます。つまりこのＩＣの場合、出力は補の値として得られます。このときアドレス入力は、$AD_D = 0$、$AD_C = 0$、$AD_B = 1$、$AD_A = 0$ です。

保持状態とは、このＩＣが動作しない状態のことを示しており、このときの出力は全て 1 の状態のままとなります。

第 13 章

インターフェイス

ここではデジタル回路と、その他の装置とを接続するための手法について述べます。

人間の感覚器官は、デジタル信号ではなくアナログ信号のみ関知することが出来ますのでデジタル信号そのままでは、人は理解することが出来ません。そこでデジタルからアナログへの変換が必要になります。これらを取り扱う理論は、非常に多くの文献があり、この様に単なる章として記述できるような内容ではありませんが、ここではこれらの装置について簡単に概要を説明します。

また同じデジタルでも扱っている信号の大きさが異なる場合、あるいは異なったデバイスを用いる場合がありますので、そのときにも変換が必要になります。異なったデバイスとは BJT と CMOS などの、従来は別のＩＣチップとして用いられてきた素子が、それらの長所を補完するべく混在したＩＣとして同じチップ上に作り込まれている場合などです。

また同じＩＣ上において、距離が離れた場所へ信号を伝送する必要から、信号の減衰を防ぐために途中にバッファを挿入するような場合もありますが、ここではこの様なバッファについては取り扱いません。

13.1　素子の特徴

インターフェイスの問題を考える場合、インターフェイスが必要であるかどうかは、使っている素子がどの様な種類の素子で、その素子がどの様な特徴を持っているかを知る必要があります。

デジタル信号を取り扱う素子には、大きく MOS と BJT に分けられます。使っている素子が違いますので、これらを接続するためには、各々の特徴を考慮しなければなりません。

MOS は、ゲートのインピーダンスが非常に大きいため、ゲート側での電流の入出力はありません。つまり前段のデバイスにおいて、電流が必要な素子が使われるときには、何らかの形で別の場所から電流を供給してやる必要があります。

さらに MOS は、BJT に較べて g_m が小さいので、外部出力を駆動する十分な能力がありません。g_m が小さいと言うことは、デバイスの理論から出力インピーダンスが大きいと言うことを意味しており、小さいインピーダンスが接続されていると、そのインピーダンスに生じる信号は大きく減衰すると言うことを意味しています。

また MOS は、ON 抵抗が結構大きいので、抵抗が小さくないと困る場合には、 何らかの回路な工夫が必要になります。

BJT の場合は、エミッタバック特性が非常に壊れ易いため、大きな電圧信号 を入力することは出来ません。最近の BJT は、高周波特性を改善するために、 ベース領域は非常に薄くなっています。そのためエミッタに逆バイアス を掛けてブレークダウンさせた場合、ドーピングした不純物が拡散して元に戻らな くなる確率が非常に高くなっています。この現象が生じると、BJT の H_{FE} が低下して、最後には BJT では無く単なる抵抗へと劣化してしまいます。これは最悪の不良発生です。なぜならば使い初めのときには正常

に動作しますが、使っている内に劣化し最後には動かなくなってしまうからです。つまり製品として出荷された時点では良品であるのですが、購入者が使っている内に故障します、しかもかなり短時間で故障状態となるからです。

市販のＩＣの中には、コレクタがそのまま出力されているだけの製品も多く存在します。これはデバイスの問題と言うよりも製品を作るポリシーですが、使っている方から見ると不良品であるかのような錯覚に陥ります。特に初めてデジタルＩＣを用いて設計しようとしている人は、注意が必要です。

大きな出力電圧が必要な場合には、デジタルに使っている素子の耐圧を超える場合もありますので、注意が必要です。特に注意しなければならないのは、コイルなどの影響で電源投入時過大な電圧が発生することもありますので、定常状態での耐圧だけを考慮しないで、過渡的な状態も考えておかねばなりません。

以上のことから、次のような工夫が必要になります。

1. MOS 入力の素子を使う場合、抵抗を追加するなどして前段の電流を補わねばならない。
2. MOS の出力側に電流が必要な何らかの装置を接続する場合、駆動能力が不足であるときには、 駆動能力を備えたバッファーを挿入しなければならない。
3. 低い ON 抵抗が必要な場合、電流を増やし素子の ON 抵抗を下げるか、それでも不足である場合には別の素子を追加して ON 抵抗を下げねばならない。
4. BJT で出来ているデジタル回路の入力側がエミッタである場合、エミッタ耐圧を越えないように回路の工夫をしなければならない。
5. コレクタ出力になっている場合には、回路が ON 状態になるように十分大きな値の抵抗を VCC との間に挿入する必要がある。

6. 過渡状態において問題の発生がないことを確認する。

13.2　具体例

ここではインターフェイスの問題として、よく出てくる代表的な例について述べておきます。

TTL と CMOS

TTL の入力はエミッタオープンの場合が多い。先に述べたようにエミッタバック は耐圧が小さいことと、エミッタバック電流は素子を破壊してしまいますので、ブレークダウンしないような入力に設定しなければなりません。

TTL の出力は、コレクタオープンとなっている場合があります。よって TTL の出力に何も接続されていない場合、出力のＢＪＴが OFF のとき、出力電圧は定まらない状態となり、ON 状態のときはＢＪＴがベース・エミッタ間の単なるダイオードとなるため、TTL の中で大きな電流が流れてしまいます。この様なことを防ぐには、通常 V_{CC} と TTL 出力の間に大きな値の抵抗を挿入します。この抵抗の値は大きいほど ON 状態の電圧は低下するがその分より強く飽和状態に入ってしまうので、次の段に必要な電圧を考慮して決める必要があります。

CMOS 入力は、MOS のゲートがそのまま外部へ出ている場合が多く存在します。MOS のゲートは静電破壊に対して非常に弱いので、TTL を接続する場合には静電破壊に注意しなければなりません。

ゲートがそのまま出ているときには、ゲート電圧は決まっていませんので、CMOS に接続される TTL から供給しなければなりません。TTL がコレクタオープンのときは、V_{CC} との間に抵抗を挿入します。

CMOS 出力は、ソース出力の場合が多いのですが、ソースのインピーダンスはかなり大きい値を持っています。そのため小さいインピーダンスを持った TTL のデバイスを多数接続する場合は、駆動することが出来ません。このときにはよりインピーダンスが小さいバッファを CMOS の後ろに接続しなければなりません。

表示装置

表示装置は、比較的大きな電圧と、電流が必要です。よって表示装置に必要な電圧や電流がデジタル回路だけで賄えない場合には、デジタル回路と表示装置の間にバッファーを挿入しなければなりません。

表示装置は一般的に、電流が流れ始めると急速に抵抗値が低下します。何も工夫をしない状態では、抵抗値の減少によって非常に大きな電流が流れ始め、最終的にデジタル回路が破損してしまいます。このことを防ぐための最も簡単な手段としては、表示装置と直列に抵抗を挿入する方法です。

例えば、表示装置として LED を使った場合、抵抗の値を R とし、流れる電流を I、電源電圧を V_{CC} とすればデジタル回路の出力電圧をゼロとしたときの LED に加わる電圧は、$V_{LED} = V_{CC} - RI$ と与えられます。この式を LED のグラフの上に重ねて描くと図 13.1 となります。この二つの線が交わっているところが実際に流れる電流ですので、この電流値が LED に必要な規定の値になるように抵抗値 R を決めさえすればよいのです。

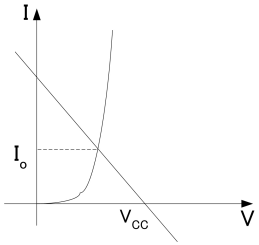


図 13.1　保護抵抗

第 14 章

ＡＤ/ＤＡ変換器

ＡＤ/ＤＡ変換器は、我々人間の視覚・聴覚・触覚とデジタル信号とを結びつけるための重要な電子回路 であり、この発展は近年目を見張るような発展をしてきており、その重要性から別の章をもうけて説明することにしました。

まず始めにＡＤ/ＤＡ変換器において重要な、演算増幅器（オペアンプ：Operational Amplifier）の説明から 始め、次にＤＡ変換器、最後にＡＤ変換器の説明を行います。

14.1　オペアンプ

増幅器には、様々な種類があり、その分類の仕方も色々考えられています。それらの中で入力が電圧で出力が電流となる電子回路のことを電圧制御電流源（Voltage-controlled Current Source：ＶＣＣＳ）と呼んでいますが、オペアンプはこの分類の中にはいります。オペアンプは、OPamp（OP アンプ）とも呼ばれ、増幅器の一種であるが、次のような特徴を持っています。

1. 利得が無限大
2. 入力インピーダンスが無限大

3. 出力インピーダンスがゼロ

この条件を満足する、オペアンプの回路図を図 14.1、図 14.2 に示しておきます。図 14.1 は一入力のオペアンプ、図 14.2 は二入力のオペアンプです。二入力オペアンプの場合プラス側入力端子の電圧が上昇したとき、出力電流が増え、マイナス側入力端子の電圧が上昇したとき、出力電流が減少するように働きます。マイナス側入力を反転入力、プラス側の入力を非反転入力と呼んで、各々図 14.1、図 14.2 のようにマイナス記号、プラス記号を付けるのが普通の表記法です。

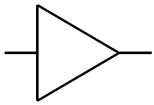


図 14.1　一入力アンプ

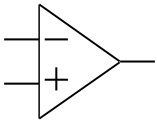


図 14.2　二入力アンプ

オペアンプの特性をそのままの形で実現することは出来ませんが、近似的な特性を持った回路は実現することが出来ます。

このオペアンプとは、日本語でも演算増幅器と呼ばれていますように、様々な演算を行う回路を作成することが出来ます。そのうち重要な回路についてあげておきます。

インバータ　オペアンプは元々増幅器の理想型であるので、当然のことながらインバータとして動作させることが出来ます。その回路図を図 14.3 に示しておきます。

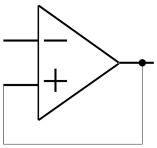


図 14.3 インバータ

正相増幅器 この回路は入力信号と同位相の回路を出力する回路であり、その増幅率は (14.1) 式によって与えられます。

反転増幅器 この回路は入力信号と逆位相の回路を出力する回路であり、その増幅率は (14.2) 式によって与えられます。

$$G_{positive} = 1 + \frac{Z_1}{Z_2} \tag{14.1}$$

$$G_{negative} = -\frac{Z_2}{Z_1} \tag{14.2}$$

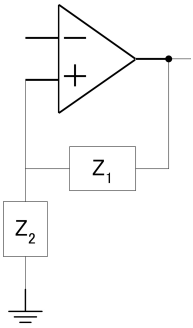


図 14.4 正相増幅器

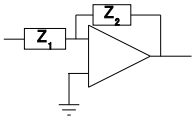


図 14.5 反転増幅器

加算器 反転増幅器は、図 14.6 のような構成にすれば、簡単に加算器とすることが出来る。二つの入力加算についての例を示しておきました。

このときの利得は、次の (14.3) 式のように与えられます。

$$G_{add} = -\left(\frac{Z_3}{Z_1} + \frac{Z_3}{Z_2}\right) \tag{14.3}$$

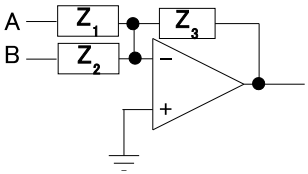


図 14.6　加算器

14.2　DA 変換

DA 変換器は D2A あるいは DAC と書かれることもあります。デジタル信号をアナログ信号へ変換する装置です。4 ビットの DA 変換器の真理値表を表 14.1 に示します。この表において A , B , C , D はデジタル入力 V_{out} はアナログ出力を示しています。この表を見ると、デジタル入力の A , B , C , D に対して各々 1 , 2 , 4 , 8 の重み付けをした値が出力のアナログ信号になっていることが分かります。このことからデジタルの各入力に対して重み付け、つまり各値に重みを掛け合わせた後、加えあわせることによってアナログ出力が得られることが分かります。

	デジタル				アナログ		デジタル				アナログ
番号	D	C	B	A	V_{out}	番号	D	C	B	A	V_{out}
1	0	0	0	0	0	9	1	0	0	0	8
2	0	0	0	1	1	10	1	0	0	1	9
3	0	0	1	0	2	11	1	0	1	0	10
4	0	0	1	1	3	12	1	0	1	1	11
5	0	1	0	0	4	13	1	1	0	0	12
6	0	1	0	1	5	14	1	1	0	1	13
7	0	1	1	0	6	15	1	1	1	0	14
8	0	1	1	1	7	16	1	1	1	1	15

表 14.1 DA 変換真理値表

具体的な回路としては、過去様々な回路が考案されています。主な DA 変換器として、次の変換器があります。

14.2.1 乗算型 DA 変換器

入力の電圧もしくは電流と、入力されるデジタル信号との掛け算によって出力を得る方法であり、簡単な回路図を図 14.7 に示しておきます。

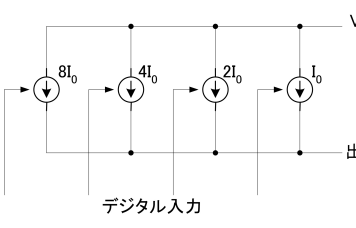


図 14.7 電流乗算型 DA 変換器

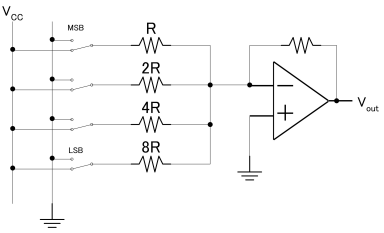


図 14.8 電圧乗算型 DA 変換器

図 14.7 図 14.8 共にデジタル信号は、各々の電流源あるいは電圧源を動作させるために用いられます。そのため電流源は、必要な電流を得るために各々必要な重みを持っていなければならないし、電圧源の場合には重みの実

現はその後に接続された抵抗とオペアンプによって得られるようにしなければならなりません。

またこの様にして得られたアナログ信号は、段階的に変化する信号として得られてしまいますので、滑らかな信号へと変換する必要があります。これらの階段的に変化する成分は、本来のアナログ信号と比較して周波数が高いところに通常は存在していますので、これらの成分を取り除くために、各々の回路の出力側に低域通過フィルタを接続しておく必要があります。

これらの DA 変換器は、電流源の電流精度、あるいは抵抗の値の精度を非常に高く取る必要があります。そうしなければアナログ信号を忠実に再現できなくなってしまいます。そこでこの問題を解決しようとして考案されたのが、次に述べるビットストリーム型 DA 変換器です。

14.2.2　ビットストリーム型 DA 変換器

ビットストリーム型 DA 変換器の中にはデルタ変調 が用いられています。まず始めにこのデルタ変調の説明から行います。デルタ変調のデルタ (Delta) とは、いわゆる差のことを指しています。その言葉の意味からも分かりますように、信号振幅の差を利用して変調する装置のことを指しています。デルタ変調ついでにデルタ復調は、次の図 14.9 に示すように 与えられます。

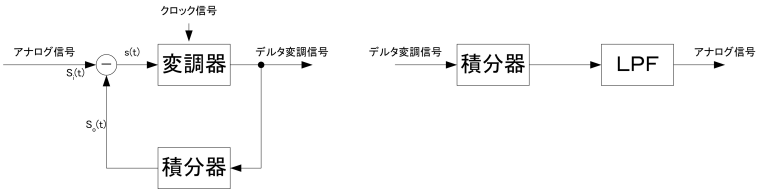


図 14.9　デルタ変復調ブロック

デルタ変調は、入力信号 $S_i(t)$ と出力を積分した信号 $S_0(t)$ が、次の式

(14.4) で示される差信号 $s(t)$ として変調器に入力されます。この変調器は、差信号 $s(t)$ が正の値となるときには、正のパルスを出力として出し、負の値場合には負のパルスが出力されます。また差信号 $s(t)$ がゼロの場合には、正負のパルスが出力されます。

$$s(t) = S_i(t) - S_o(t) \tag{14.4}$$

この様なパルスの出力を行うことによって、出力側を積分しますと入力信号に応じた階段波形が得られることになります。デルタ変調は、入力信号の差の値に応じてパルスを出しますので、直流信号のように変化のない信号は取り扱うことは出来ません。

ビットストリーム DA 変換器は、このデルタ変調を用い て行われ、そのブロック図は図 14.10 として与えられます。

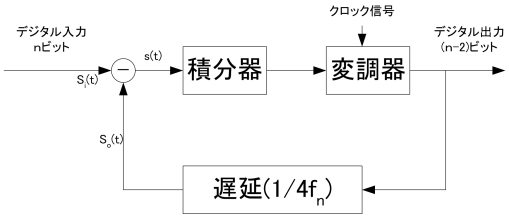


図 14.10　ビットストリーム DA 変換器

14.3　AD 変換

ＡＤ変換器は、A2D あるいは ADC と呼ばれることもあります。アナログ信号をデジタル信号へと変換する装置です。

ＡＤ変換器は、一般的に DA 変換器に較べて動作が遅く、しかも設計が大変です。その理由はアナログ信号を正確にデジタルの対応する値にあわせていく必要があるからです。

次に現在までに考案された主なＡＤ変換器を次に示します。

14.3.1　逐次近似ＡＤ変換器

この手法は、基準電圧を 2 分の 1 ずつ変化させ、その電圧よりも大きければ 1 小さければ 0 を最上ビット側から順次決めていく方式です。

14.3.2　デュアルスロープＡＤ変換器

デュアルスロープＡＤ変換器は、電荷がゼロの状態になっていますある値の容量を決められた時間だけ入力の電圧で充電を行い、次に容量を別の電圧源を用いて放電させます。この容量に蓄積されていた電荷がゼロになるまでの時間をカウンタで計測し、このカウンタの計測をデジタル信号として読みとる方法です。

電荷の充放電を用いますので、変換の速度はかなり遅いのですが、素子のペア性などを利用することが出来ますので、変換の精度としては優れた性能を得ることが出来ます。

14.3.3　フラッシュＡＤ変換器

図 14.11 に示すようにフラッシュＡＤ変換器は、入力のアナログ信号電圧と基準電圧とを比較して、アナログ信号をデジタル信号へと変換する方法であり、回路図から分かりますように非常に高速に動作させることが出来ます。しかし基準電圧を作る必要があり、図 14.11 のような回路の場合非常に高精度の抵抗の精度が要求されます。このためビット数の大きなフラッシュＡＤ変換器を実現することは困難です。

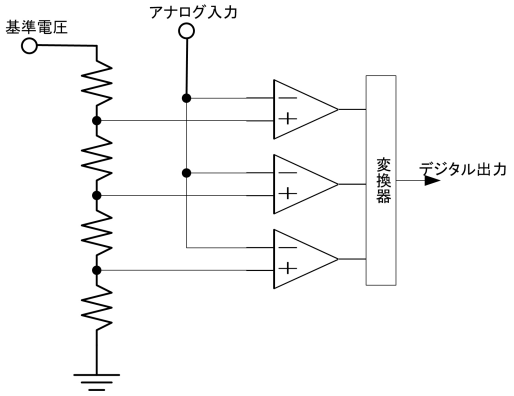


図 14.11　フラッシュＡＤ変換器

14.3.4　シグマデルタＡＤ変換器

シグマデルタ変換器は、デルタ変換器を改良した回路方式で、直流信号も伝送することが可能です。この方式はただ単に時間前後の信号の差を取るのではなく出力信号との差信号を積分した後にデジタル化を行う方式です。このシグマデルタ変換器を図 14.12 に示しておきます。

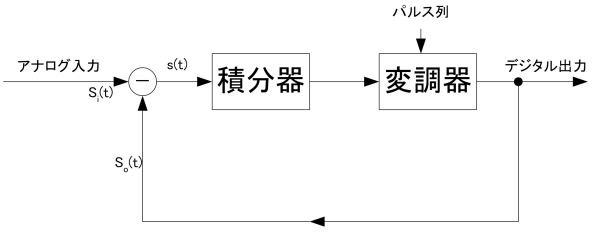


図 14.12　シグマデルタＡＤ変換器

第 15 章

デジタル・フィルタ

デジタル・フィルタは、Ｚ変換と呼ばれるフーリエ変換をデジタルへ 拡張した数学が用いられます。Ｚ変換については、付録に簡単に述べておきます。

デジタル･フィルタは､FIR(Finite Impulse Responce) と IIR(Infinite Impulse Responce) とがあります。日本語で FIR は有限長インパルス応答、 IIR は無限長インパルス応答と呼ばれています。

FIR フィルタは、出力が有限個の入力に依存したフィルタであるためこの様な名前が付けられています。これに対して IIR は、過去の全ての入力に依存して出力が決まるフィルタのことを示しています。

次にこれらのフィルタについて、もう少し詳しく説明していくことにします。

15.1　FIR フィルタ

具体的な例を考えながら説明をしていきます。次のようなブロック図を考えます。

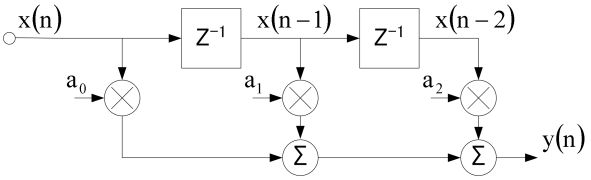


図 15.1　FIR フィルタの例

このブロック図から、次の式が得られます。

$$y(n) = a_0 x(n) + a_1 x(n-1) + a_2 x(n-2) \tag{15.1}$$

この様に出力が入力の式だけで与えられる形式のフィルタを FIR フィルタと呼んでいます。この中で a の三つの値は、フィルタの形を決める係数です。

z^{-1} は、z 変換から出てきた変数です。この値を逆 z 変換してみますと

$$\frac{1}{2\pi j}\oint z^{-1} z^{n-1} dz = \frac{1}{2\pi j}\oint z^{n-2} dz$$

この関数は、$n = 1$ のところで特異点を持ちます。値が 1 というのは、デジタル信号における最少のサンプリング間隔を示しています。つまり z^{-1} は、最少サンプリング間隔の時間遅れることを意味しています。このことを利用して、次の例について考えてみます。

またここではデジタルフィルタをどの様に構成するかという問題については、考えません。このテーマについては、デジタル・フィルタの専門書をご覧下さい。

例

次に示す株価データを、上で与えたフィルタに通したときどの様になるか考えてみます。ただしフィルタの係数として、次の値を取るものとします。

$a_0 = 0.25$, $a_1 = 0.50$, $a_2 = 0.25$

また日付のない日の株価は、ゼロであるとします。

曜日	$x(n)$	株価
月曜日	$x(0)$	20
火曜日	$x(1)$	20
水曜日	$x(2)$	15
木曜日	$x(3)$	12
金曜日	$x(4)$	40
土曜日	$x(5)$	30

表 15.1　株価変動

この表をグラフで示しますと、次のように得られます。

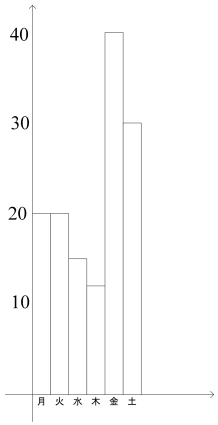


図 15.2　株価変動グラフ

これから、次の式が得られます。

$$
\begin{aligned}
y(0) &= 0.25\times 20 + 0.50\times 0 + 0.25\times 0 = 5\\
y(1) &= 0.25\times 20 + 0.50\times 20 + 0.25\times 0 = 15\\
y(2) &= 0.25\times 15 + 0.50\times 20 + 0.25\times 20 = 18.75\\
y(3) &= 0.25\times 12 + 0.50\times 15 + 0.25\times 20 = 15.5\\
y(4) &= 0.25\times 40 + 0.50\times 12 + 0.25\times 15 = 19.75\\
y(5) &= 0.25\times 30 + 0.50\times 40 + 0.25\times 12 = 30.5
\end{aligned}
$$

この結果をグラフで示しますと、次のように求められます。このグラフ共とのグラフ図 15.2 を較べますと、極端なでこぼこが無くなり低域通貨フィルタとなっていることが分かります。

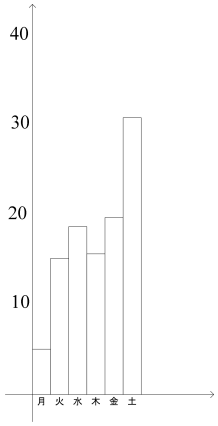


図 15.3　フィルタ通過後のグラフ

FIR フィルタは、過去の入力にある計数を書けた式によって表現することが出来ます。また次の関係が成立しますので、

$$X(z)z^{-k} = \frac{1}{2\pi j}\oint x(n-k)z^{-n}dz$$

この関係式かを考えますと、FIR フィルタの伝達関数は、必ず z^{-k} の関数となります。この関数はゼロ以外の極点がありませんので、必ず安定な関数となります。このことから FIR フィルタは全て安定なフィルタとなります。この性質は、アナログ・フィルタにはない性質です。

15.2 IIR フィルタ

次のブロックに示すような入力側が元の入力側に戻る回路を考えてみます。

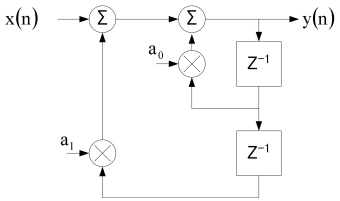


図 15.4 IIR フィルタの例

このフィルタの式は、次のように与えられます。

$$y(n) = x(n) + a_0 y(n-1) + a_1 y(n-2)$$

この式から分かりますように、IIR フィルタは、過去の出力の値を用いて記述します。このため場合によっては発振することもありますので、注意が必要です。

実際のフィルタを設計する場合、FIR フィルタと IIR フィルタとをミックスした様なフィルタを使う場合が多いようです。

これらのフィルタについての詳細は、デジタルフィルタの専門書を参照して下さい。

第 16 章

ディジタルの数学

アナログ信号を取集時に時間空間だけでは無く周波数空間を考えると、時間空間では扱うことが難しかった色々なことが簡単に扱えるようになりました。このように周波数空間においてディジタルの世界も扱えるようにしたいというのが Z 変換と呼ばれて開発されました。

16.1　デルタ関数 δ について

ディジタルの数学を説明する前に重要なデルタ関数 δ[*1] について述べておきます。

デルタ関数は超関数とも呼ばれる数の一種で、次のような性質を持っています。証明などについては参考文献 [8] を御覧下さい。

16.1.1　デルタ関数の定理

デルタ関数には様々な定理が存在しますが、数学書ではないのでここでは証明は先程述べました参考文献にまかせ結果だけを述べておきます。

定理を述べる前にデルタ関数は幅がゼロで、高さが 1 であり、面積も 1 の

[*1] ディラックのデルタ関数とも言われます。

関数となる特殊な関数です。次にこの関数の色々な定理について述べていきます。

ステップ関数の微分　高さが 1 その後この値が続き、直角に立ち上がる関数をステップ関数 $U(t)$ と呼んでいます。この関数との間には、次の関係があります。

$$\delta(t) = \frac{dU(t)}{dt} \tag{16.1}$$

面積は 1　デルタ関数は次の式を満足します。

$$\int_{-\infty}^{\infty} \delta(t) = 1 \tag{16.2}$$

デルタ関数は偶関数　次の式が成り立ちます。

$$\delta(-t) = \delta(t) \tag{16.3}$$

ある関数とデルタ関数の席による積分　ここで t_0 は正の任意の値とします。

$$\int_{-\infty}^{\infty} f(t)\delta(t-t_0) = f(t_0) \tag{16.4}$$

デルタ関数の微分は奇関数　次の式が成立します。

$$\delta(-t) = -\delta(t) \tag{16.5}$$

ある関数と微分デルタ関数の積の積分

$$\int_{-\infty}^{\infty} f(t)\frac{d\delta(t)}{dt} = -\frac{df(t)}{dt}\bigg|_{t=0} \tag{16.6}$$

16.2　定義から各種定理へ

アナログ信号を等間隔の時間で飛び飛びの値を抽出するとします。その感覚時間 T として n 個目の信号のように番号付けします。ただし $n = 0$ は時間ゼロとなるように決めます。これにより信号 $f(x)$ の n 番目の信号は $f(n)$

と表されることになります。その結果飛び飛びの信号は時間ゼロを含むと、次のように表現できます。

$$f(0)+f(T)+f(2T)+f(3T)+\cdots=\sum_{n=0}^{\infty}f(nT) \tag{16.7}$$

このままでは数学的な計算などは出来ませんので、次のようなラプラス変換からいわゆる z 変換と呼ばれる変換を考えていきます。

16.2.1 ラプラス変換から Z 変換へ

まず片側ラプラス変換は、次の式で定義されています。

$$\mathscr{L}[f(t)]=\int_0^{\infty}f(t)\exp(-st)dt$$

ここで (A.1) 式を次のように書き直します。四季を書き直す際にディラックのデルタ関数 δt を使います。デルタ関数というのは幅がゼロである、超関数と言われている関数です。この関数を用いると上の式の $f(t)$ に (A.1) 式を使うと上の片側ラプラス変換は、次のようになります。

$$\begin{aligned}\mathscr{L}\sum_{n=0}^{\infty}f(t)\delta(t-nT)&=\int_0^{\infty}\sum_{n=0}^{\infty}f(t)\delta(t-nT)\exp(-st)dt\\&=\sum_{n=0}^{\infty}f(nT)\exp(-snT)\end{aligned}$$

ここで

$$z=\exp sT \tag{16.8}$$

と置きますと 16.10 式は、次のようになります。

$$\mathscr{Z}[f(t)]=\sum_{n=0}^{\infty}f(n)z^{-n} \tag{16.9}$$

この式の中で $\mathscr{L}$ の代わりに $\mathscr{Z}$ を用いていることと、関数 f の変数として時間 T を省略していることに注意して下さい。ここで $z=\exp(sT)$ と定義

していますが s は複素平面全体の任意の点ですので、z も複素平面全体の点を示すことになります。そこで上の式を z で書きそれを Fn と書きますと、次のように書き換えられます。この式を見て分かることは時間間隔 T が省略されているだけで、複素空間が別のふくす空間に変換されているだけで、ラプラス変換の性質自体はそのまま引き継がれていることになります。

$$Fn = \sum_{0}^{\infty} f(n) z^{-n} \tag{16.10}$$

n は整数の範囲で任意の値を取ることが出来ます。例えば $n-1$ その他適当なマイナスの値を取ることが出来ます。そうすると定義式は正の時間から出発しているのに、マイナス時間も許されるのかという疑問が生じます。これは別にマイナスの時間を許しているのでは無く、自体はその以前から始まって回路の中でメモリなど過去蓄積された現象を扱うために必要となるため、まるでマイナス時間を扱っているかのように思えるだけで、全ての現象に記憶という要素が入ると過去の記憶がどのように将来現れるかということを扱うためには必要なことなのです。

例えば $F(n-1)$ は、次のように決められています。

$$\begin{aligned} F(n-1) &= f(-1) + f(0)z^{-1} + f(1)z^{-2} + f(2)z^{-3} + \cdots \\ &= f(-1) + z^{-1}[f(0) + f(1)z^{-1} + f(2)z^{-2} + \cdots] \\ &= f(-1) + z^{-1}F(n) \end{aligned} \tag{16.11}$$

この式から $f(-1)$ の項が生じることになります。これは $t = (-1)T$ 時間前の信号ですので、過去の信号が現在の信号に影響を与えていることになります。これはディジタル信号の場合先程述べましたように記憶を持っていることがあるので、過去の現象を表現する必要があるからです。$f(n-m)$ の場合は、次のように与えられています。

$$Fn-m = z^{-m}F(n) + \left[z^{-m+1}f(-1) + z^{-m+2}f(-2) + \cdots + f(-m)\right] \tag{16.12}$$

この式で注意しないといけないのは、$t = 0$ より前では z^{-1} の係数の付け方が異なることです。またこのように過去 $t = mT$ 時間前までの記憶が現在

に使われることになります。

16.2.2 逆ラプラス変換から逆 Z 変換へ

逆ラプラス変換は、次のように定義されています。

$$\mathscr{L}^{-1}F(s) = \frac{1}{2\pi j}\int_{-\infty}^{\infty} F(s)\exp(st)ds$$

ここで z は、複素数です。x は、信号を表現している連続関数ですが、$x(n)$ は、$t = n$ のところでの x の値を示しています。このことから $x(n)$ は、$t = n$ のところでの離散値の列を示していることになります。

フーリエ変換の場合と同様に、(16.10) 式は、両側 Z 変換と呼ばれます。この式に対応して、次のように定義される変換のことを、片側 Z 変換と呼んでいます。

$$X(z) = \sum_{n=0}^{\infty} x(n)z^{-n} \tag{16.13}$$

また逆 z 変換は、次の式として定義されています。

$$x(n) = \frac{1}{2\pi j}\oint X(z)z^{n-1}dz \tag{16.14}$$

ただし積分領域は、原点を囲む線です。

16.2.3 フーリエ変換との関係

Z 変換は、フーリエ変換と密接に関係しています。複素数 z を、次のように極座標で表現してみます。

$$z = re^{j\omega} \tag{16.15}$$

この様に置きますと、(16.10) 式は、次のように変形することが出来ます。

$$\begin{aligned} X(j\omega) &= \sum_{n=-\infty}^{\infty} x(n) \left[re^{j\omega} \right]^{-n} \\ &= \sum_{n=-\infty}^{\infty} \left[x(n) r^{-n} \right] e^{-j\omega n} \end{aligned} \tag{16.16}$$

この最後の式を見ますと、フーリエ変換そのものであることが分かります。特に $r = 1$ としますと、フーリエ変換となりますので、Z 変換はフーリエ変換の拡張となっていることが分かります。

付録 A

Z 変換

ディジタル信号で用いられている z 変換について記述しています。

A.1　z 変換の定義

片側ディジタル変換は次の式によって定義されています。

$$\begin{aligned}\mathscr{Z}[f(t)] &= F(z) \\ &= f(0)+f(1)z^{-1}+f(2)z^{-2}+f(3)z^{-3}+\cdots \\ &= \sum_{n=0}^{\infty} f(n)z^{-n}\end{aligned} \tag{A.1}$$

これは片側 z 変換と呼ばれていますが、フーリエ変換やラプラス変換のときにもありましたように、両側 z 変換というものも考えられますが、実際の回路の問題などにおいては時間的に正の値を取り扱うことが多いので、両側 z 変換はほとんど用いられません。ここでも片側 z 変換のみを扱います。

A.2　z 変換の性質

z 変換についてもフーリエ変換やラプラス変換と同様な性質が成立します。ここでは代表的な性質を中心に述べています。

A.2.1　線形定理

線形とは $X(n) = x(n)$ および $Y(n) = y(n)$ とすると、次の式が成立することを言っています。

$$ax(n) + by(n) = aX(n) + bY(n) \tag{A.2}$$

A.2.2　移動定理

A.2.3　定数倍

A.2.4　指数倍

A.2.5　極限

A.3　畳み込み

A.3.1　ディジタル変換表

$x(n)$, $y(n)$ について、次の条件が成立する場合。

変数	z 関数	条件
$x(n)$	$X(z)$	$R_{x-} < \|z\| < R_{x+}$
$y(n)$	$Y(z)$	$R_{y-} < \|z\| < R_{y+}$

下の表が、得られます。

時間領域	周波数領域	条件
$ax(n)+by(n)$	$aX(z)+bY(z)$	$\max(R_{x-},R_{y-}<\|z\|<\min(R_{x+},R_{y+})$
$x(n+n_0)$	$e^{n_0}X(z)$	$R_{x-}<\|z\|<R_{x+}$
$a^n x(n)$	$X(a^{-1}z)$	$\|a\|R_{x-}<\|z\|<\|a\|R_{x+}$
$nx(n)$	$-z\frac{dX(z)}{dt}$	$R_{x-}<\|z\|<R_{x+}$
$x^*(n)$	$X^*(z^*)$	$R_{x-}<\|z\|<R_{x+}$
$x(-n)$	$X(1/z)$	$1/R_{x-}<\|z\|<1/R_{x+}$
$\Re[x(n)]$	$1/2[X(z)+X^*(z^*)]$	$R_{x-}<\|z\|<R_{x+}$
$\Im[x(n)]$	$1/(2j)[X(z)+X^*(z^*)]$	$R_{x-}<\|z\|<R_{x+}$
$x(n)*y(n)$	$X(z)Y(z)$	$\max(R_{x-},R_{y-}<\|z\|<\min(R_{x+},R_{y+})$
$x(n)y(n)$	$1/(2\pi j)\oint_c X(v)y\left(\frac{z}{v}\right)v^{-1}dv$	$R_{x-}R_{y-}<\|z\|<R_{x+}R_{y+}$

$x(n)*y(n)$ は、$x(n)$ と $y(n)$ のコンボリューションです。

表 A.1 Z 変換表

付録 B

周波数の分類と物理定数

ここではアナログ信号を取り扱う場合において、周波数の分類や物理定数について記述しています。

B.1　信号の分類と名称

色々な分け方が考えられますが、ここでは昔から行われている周波数の分類とその名称の付け方について述べています。

B.1.1　周波数の分類と名称

周波数は、取り扱う領域によって次の表 B.1 のように分類されています。

名称	周波数領域 $[MHz]$	波長 $[m]$
HF	$3 \sim 30$	$100 \sim 10$
VHS	$30 \sim 300$	$10 \sim 1$
UHS	$300 \sim 1{,}000$	$1 \sim 0.3$
L	$1{,}000 \sim 2{,}000$	$0.3 \sim 0.15$
S	$2{,}000 \sim 4{,}000$	$0.15 \sim 0.075$
C	$4{,}000 \sim 8{,}000$	$0.075 \sim 0.0375$
X	$8{,}000 \sim 12{,}000$	$0.0375 \sim 0.025$
Ku	$12{,}000 \sim 18{,}000$	$0.025 \sim 0.0167$
K	$18{,}000 \sim 27{,}000$	$0.0167 \sim 0.0111$
Ka	$27{,}000 \sim 40{,}000$	$0.0111 \sim 0.0075$

$c = 3.0 \times 10^8\ [m/sec]$ の場合

表 B.1 周波数の分類

B.2 物性に関する定数

ここではアナログ電気回路を学ぶ場合に必要な物理定数について述べています。

B.2.1 物理定数表

マックスウエルを取集場合に必ず必要となる、電気定数などについて主な定数について述べています。詳しくは市販の理科年表などが参考になると思います。

物理定数	値	単位
素電荷	1.60210×10^{-19}	[クーロン]
ボルツマン定数	1.38054×10^{-23}	$[JK^{-1}]$
真空の誘電率	8.854185×10^{-32}	$[F/m]$
真空の透磁率	$4\pi \times 10^{-7} \cong 1.256637 \times 10^{-6}$	$[H/m]$
絶対温度	-273.16	[℃]
シリコンの融点	1420	[℃]
禁制帯の幅	1.21	$[eV]$

表 B.2 物理定数表

B.2.2 シリコン中の拡散係数 (理科年表から抜粋)

実際に必要となる方は、集積回路などを製造する人には必須の定数ですが、回路関係の方もある程度知っておくことが望ましい定数です。不純物などの種類によって回路素子、特に半導体の信頼性などにも関わってきます。

次の式で表現される拡散式に対して必要となる定数です。詳細については物性に関する著書などを御覧ください。

$$D = D_0 \exp\left(-\frac{U}{RT}\right)$$

不純物	D_0 $[cm^2/sec]$	U $[kcal/mol]$
B	10.5	85
Al	8.0	80
Ga	3.6	81
In	16.5	90
P	10.5	85
As	0.32	82
Sb	5.6	91
Fe	6.2×10^{-3}	20
Au	1.1×10^{-3}	25.8

表 B.3 シリコン中の拡散係数

参考文献

[1] Graig Marven and Gillian Ewers 著、山口博久訳、ディジタル信号処理の基礎、丸善株式会社、１９９５年.

[2] William E. Wickes 著、篠崎寿夫・高橋宣明・大原茂之訳、集積デジタル回路の設計、東海大学出版会、１９８３年第７刷.

[3] J. R. Gibson 著、岩本洋訳、電子論理回路の基礎、 啓学出版社、１９８５年.

[4] Roger L. Tokheim 著、青木正喜訳、ディジタル回路、 オーム社、マグロウヒル大学演習、１９９５年.

[5] Alan V. Oppenheim and Ronald W. Schafer 著、伊達玄訳、ディジタル信号処理、コロナ社、１９７６年.

[6] 東光寛英、論理学入門、関書院新社、１９６４年.

[7] Carl Frederich Gauss、ガウス整数論、朝倉書店、１９９５年.

[8] Athanssions Popoulis, The Fourie Integral and its Applications, McGraw-Hill,New York,1970.

[9] Athanssions Popoulis, Circuit and Systems, CBS 出版,1980.

索引

著者紹介

新原　盛太郎（しんばら　せいたろう）

１９４８年８月	山口県徳山市に生まれる（本籍は博多区須崎）
１９７２年３月	九州大学工学部卒業
同　　年　４月	東京大学工学部　青木研究室
１９７３年４月	東京芝浦電気入社
１９８９年１０月	CADENCE 社（USA）に１年間駐在
１９９０年４月	東芝半導体事業部
２００８年４月	東京工芸大学　非常勤講師
同　　年　８月	東芝退社
２０１９年３月	東京工芸大学退任

[特許]

US ５件登録、EU ２件登録、国内特許約５０件以上

[著作]

「ダイオード/トランジスタ/FET 活用入門」　CQ 出版

「SPICE とデバイス・モデル」　CQ 出版

「間違いが多い電気知識」東京図書出版

「基礎電気回路」Amazon・電子書籍

「アナログ回路設計法」Amazon・ペーパーバック

「設計のための電気」Amazon・ペーパーバック

２０２３年（令和５年）３月　初版発行