2005年9月6日作成
2011年5月9日更新
2017年6月7日更新

３．コンピュータの基礎（３）–算術演算–
Arithmetic operation

ディジタル・コンピュータを使った計算のしかたを紹介します。普通の加減乗除の計算のことを「算術演算 arithmetic operation」と言います。ディジタル・コンピュータでは数値を二進数 (binary number) を使って表します。二進数の算術演算は，前に説明した「論理演算 logical operation」の組み合わせで実現できます。

３–１　論理回路による足し算　Addition with logical circuits

二進数の 1 桁は 0 か 1 の値しかとりません。この二進数での 1 桁のことを 「1 ビット bit」といいます。はじめに，「1 ビットの足し算回路」について考えます。つまり，

$0+0=0$
$0+1=1$
$1+0=1$
$1+1=2$

という４通りの場合について正しい答えを出力するための回路です。十進数 (decimal number) での 2 という数は二進数 (binary number) だと 10 と表されます。このことをはっきりとさせるために，

$(2)_d = (10)_b$

と書く場合があります。上の例を二進数で表現すると，

$0+0=00$
$0+1=01$
$1+0=01$
$1+1=10$

と書けます。

このような計算をどのような電子回路で実現できるでしょうか？おおまかには，２本の入力信号線と２本の出力信号線を備えた回路になるはずです。つまり，足し算の２つの入力を A, B という２本の入力信号線に，出力のうち「$2^0 = 1$」の桁の数字を Y_0 という出力信号線,「$2^1 = 2$」の桁の数字（繰り上がり）を Y_1 という出力信号線に割り当てます。全体としては，図 3.1 のような形になるわけです。

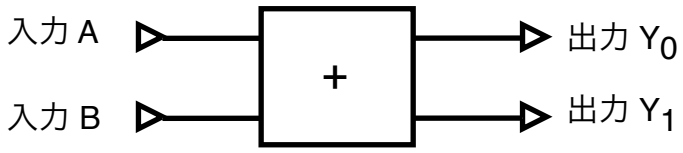

図 3.1　1 桁の足し算回路。Y_0 は 2^0 の桁の数，Y_1 は 2^1 の桁の数を出力する。

この回路の中身を考えましょう。出力 Y_0，Y_1 と入力 A, B の間には表 3.1 のような関係があります。

表 3.1　「1 ビット」の足し算回路, 2^1 の桁の出力 Y_1 と 2^0 の桁の出力 Y_0

A	B	Y_1	Y_0
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

この関係を論理式を使って書くために，次のように考えます。表 3.1 から，「出力 Y_0 が 1 になる」のは，「$A = 0$ かつ $B = 1$」の場合と，「$A = 1$ かつ $B = 0$」の場合との 2 通りあります。これを式で表せば,

$$Y_0 = \left(\overline{A} \wedge B\right) \vee \left(A \wedge \overline{B}\right)$$

となります。この関係は，第 1 回の論理ゲートのところに出てきた「排他的論理和 XOR」と同じです。つまり，XOR ゲートを使えば一桁の足し算の $2^0 = 1$ の桁を得る回路になるわけです。

つぎに，出力 Y_1 と入力 A, B の間の関係は,

$$Y_1 = A \wedge B$$

となります。この関係は，「論理積 AND」そのものです。つまり，AND ゲートを使えば 1 桁の足し算の繰り上がりの部分，$2^1 = 2$ の桁を得る回路になるわけです。

これらのことをまとめれば，1 ビットの足し算をするための電子回路としては全体として, 図 3.2 のような構成にすれば良いわけです。この「1 ビットの足し算回路」は半加算器 half adder （ハーフ・アダー）とも呼ばれます。

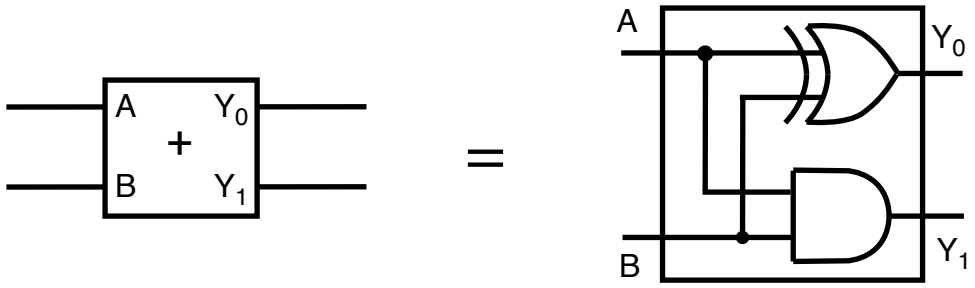

図 3.2　「1 ビット」の足し算回路 (2)（半加算器）

XOR ゲートと AND ゲートの組み合わせで実現できる。

次に，任意の桁数の足し算回路について考えます。足し合わせる２つの数が両方とも２進数で N 桁だとします。つまり，２つの数 A, B がそれぞれ,

$$A_{N-1}A_{N-2}\cdots A_j\cdots A_1A_0$$

$$B_{N-1}B_{N-2}\cdots B_j\cdots B_1B_0$$

のように表されるとします。これらを足し合わせた数 Y は，最大でも $N+1$ 桁にしかなりません。これを

$$Y_NY_{N-1}\cdots Y_j\cdots Y_1Y_0$$

と書くことにします。

足し算の結果の 2^j の桁の数字 Y_j は，A_j と B_j，それと 2^{j-1} の桁の繰り上がり carry （キャリー）があるかどうかによって決まります。2^j の桁の入力を A, B，2^{j-1} の桁の繰り上がりを C，2^j の桁の出力を Y_0，2^j の桁の繰り上がりを Y_1 と表すことにすれば，各桁について表 3.3 で表されるような「３入力１ビット足し算回路」を使えば良いことがわかります。

表 3.3　３入力１ビット足し算回路

A	B	C	Y_0	Y_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

「２入力１ビット足し算回路」を図 3.2 の記号で表すことにすれば，これを使って「３入力１ビット足し算回路」を図 3.3 のように組み立てることができます。この回路のことは全加算器 full adder （フル・アダー）とも呼ばれます。

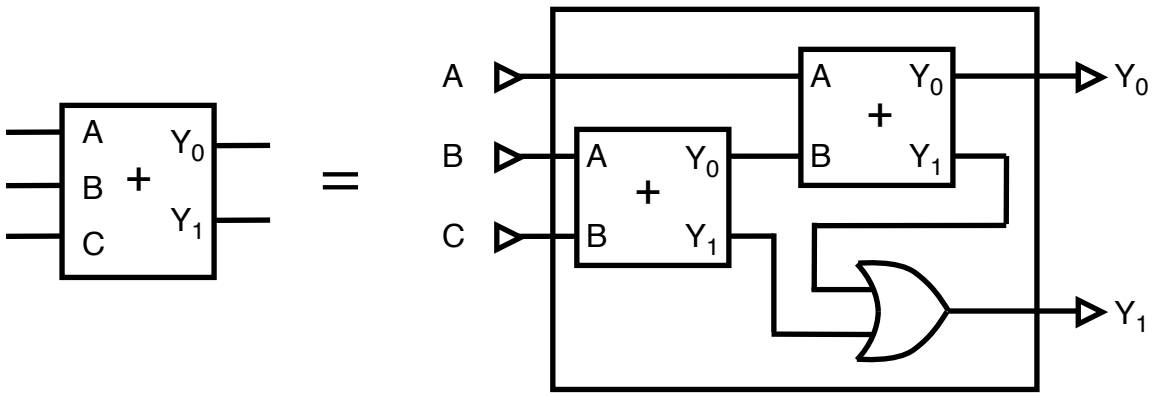

図 3.3　「３入力１ビット足し算回路」（全加算器）

この全加算器を必要な数だけ集めれば，任意のビット数の算術和を求める回路が作られます（図 3.4）。

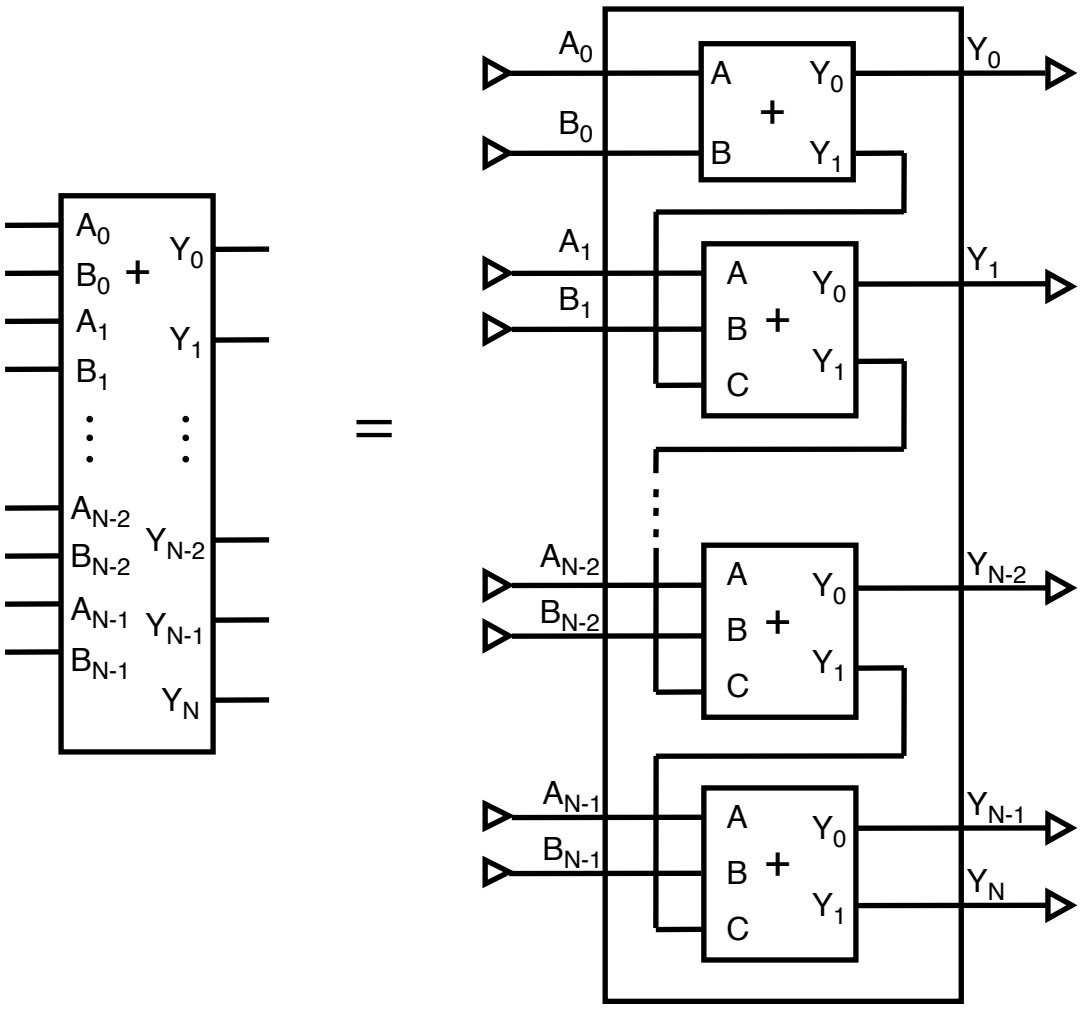

図 3.4　「２入力 N ビット足し算回路」

3－2　論理回路による引き算
Subtraction with logical circuits

引き算の結果は負（マイナス）になることもあります。一般的なディジタルコンピュータでは，負の整数を表現するために「補数表現」という方法を使います。たとえば 8 ビットの２進数で整数を表現する際に最上位ビットが「0」のときは $(00000000)_b$=0, $(00000001)_b$=1，$(00000010)_b$=2，…，$(01111111)_b$=127 のように普通に値を対応させますが，最上位ビットが「1」のときは，$(10000000)_b$ = -128，$(10000001)_b$=-127，$(11111111)_b$=-1 のように対応させます。このようにすれば，結果が -128 から 127 の範囲におさまるような足し算は，負の数が含まれていたとしても前節の足し算回路をそのまま使って計算できます。

例えば，「 9 – 5」の計算をするときに，「9 + (–5)」と見なせば足し算になります。

$9 = (00001001)_b$

$-5 = (11111011)_b$

という表現に対して，二進数での加算を行えば

$$\begin{array}{rcccccccc} & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ +) & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ \hline 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{array}$$

つまり，$(00001001)_b + (11111011)_b = (100000100)_b$ ですが，繰り上がり（キャリー）に相当する最上位ビットを無視すれば $(00000100)_b = 4$ となり，「9 – 5 = 4」という結果が得られます。

補数表現を用いた場合には，有効なビット数によってその意味する値が変わるということに注意する必要があります。たとえば，「11111111」は 8 ビットでは -1 という数を表しますが，16 ビットでは 255 という数を表します。

参考のために，「２入力 1 ビット引き算回路」について考えます。つまり，

0 − 0 = 0

0 − 1 = −1

1 − 0 = 1

1 − 1 = 0

という 4 通りの場合について正しい答えを出力するための回路です。10 進数での 「-1」という数を，2 ビットの補数表現で 11 と表すことにします。この上位ビットは「繰り下がり」を表すと考えます。この場合，上の例は，

0 − 0 = 00

0 − 1 = 11

1 − 0 = 01

1 − 1 = 00

と書けます。まとめれば表 3.4 のようになります。

表 3.4　２入力１ビット引き算回路。入力 A，B，出力 Y_0 と繰り下がり Y_1

A	B	Y_0	Y_1
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

また，このような出力を実現する回路を図 3.5 の記号で表すことにします。

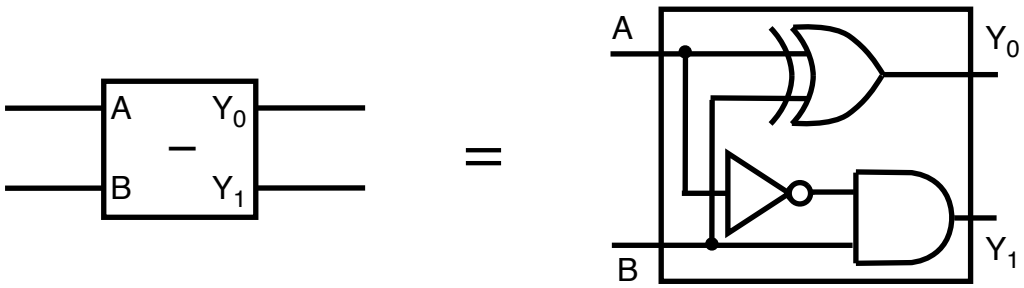

図 3.5　「２入力１ビット引き算回路」

さらに，繰り下がりの入力まで考慮して，３つの１ビット入力 A, B, C から $(A - B - C)$ の結果を出力するような３入力１ビット引き算回路を考えることにします（表 3.5）。 $0 - 1 - 1 = -2$ ですが，10 進数での「–2」という数は２ビットの補数表現では 「10」と表されます。

表 3.5　３入力１ビット引き算回路。

2^j の桁の入力 A，B と 2^{j-1} の桁の繰り下がり C，

2^j の桁の出力 Y_0 と繰り下がり Y_1

A	B	C	Y_0	Y_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

図 3.5 の「２入力１ビット引き算回路」から「３入力１ビット引き算回路」を図 3.6 のように組み立てることができます。

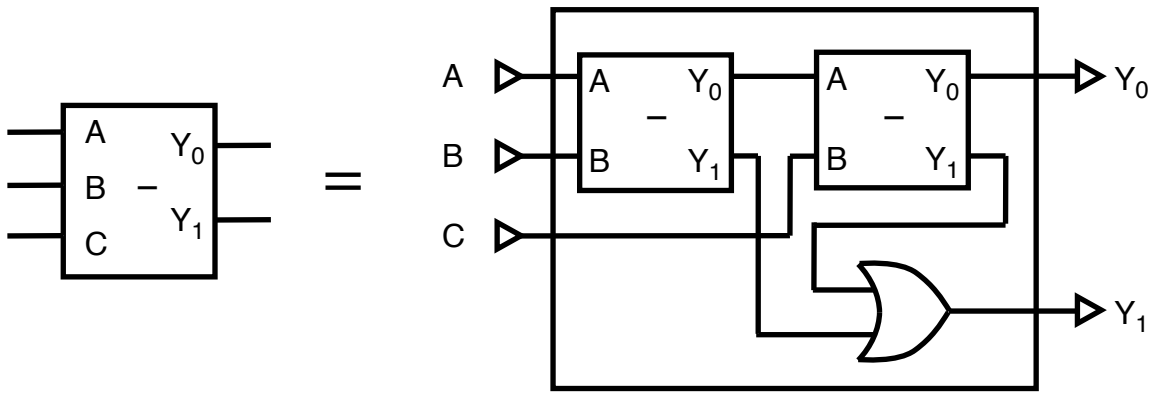

図 3.6　「３入力１ビット引き算回路」

この「３入力１ビット足し算回路を」を必要な数だけ集めれば，任意のビット数の算術的な差を求める回路が作られます（図 3.7）。

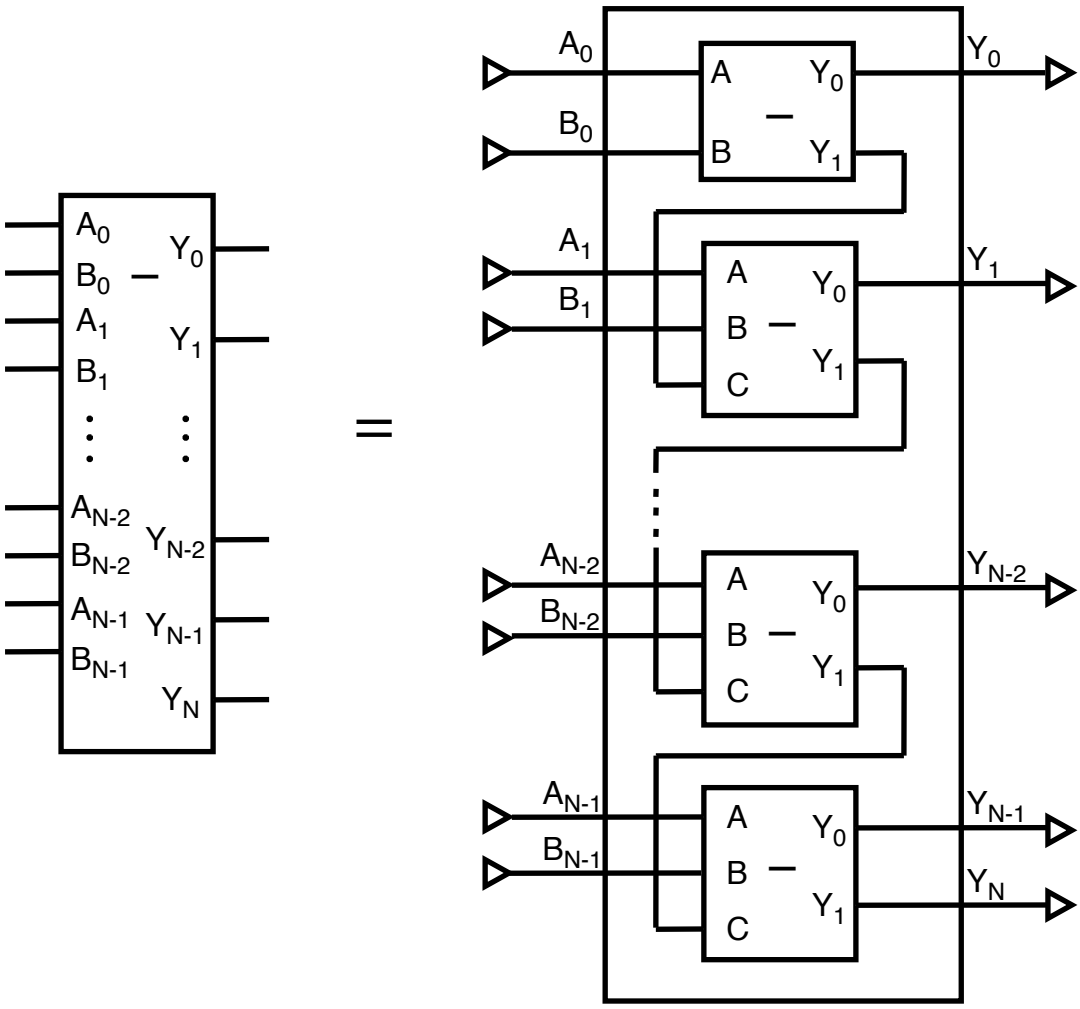

図 3.7　「２入力Ｎビット引き算回路」

３−３　論理回路によるかけ算，割り算
Multiplication & Division with logical circuits

日本の初等教育では，かけ算を九九（くく）を使って計算する方法を学びます。これは十進法で数を表すからで，数を二進法で表すことにすれば $0\times0=0$，$0\times1=0$，$1\times0=0$，

$1\times1=1$ だけ憶えれば良いことになります。後は「桁をずらす」ということと「和をとる」ということができればよいので，電子回路を使って計算することは容易です。

「九九を憶えなければ，かけ算ができない」と思い込んでいる人もいるかもしれませんが，九九を知らなくてもかけ算はできます。別名もあるようですが，「ロシア農民のかけ算」として知られている方法が有名です。

たとえば，19×25 という計算をしてみます。はじめに 19 と 25 を横に並べて書いて，次の行には19 を 2 で割った数 9（あまりは無視する；以下同じ）と 25 に 2 をかけた数 （ある

いは 25+25）50 を並べて書きます。その次の行には 9 を 2 で割った数 4 と 50 に 2 をかけた数 100 を書きます。どんどん繰り返していって，左側の数が 1 になるまで繰り返します。最後に，左側の数が奇数となっている行の右側の数を足してやればかけ算の答えになる，という方法です。

１９	○	２５
９	○	５０
４		１００
２		２００
１	○	４００

２５＋５０＋４００＝４７５

ここで，19 を 2 で割っていって奇数かどうかを調べる操作は，二進法表記を下位ビットから順に求めていくことと同じです。つまり，十進法での 19 は二進法では 10011 となります。また，このことは，$19 = 2^4 + 2^1 + 2^0$ と同じ意味です。ロシア農民のかけ算では，下の式で表されるような方法で計算をしていると考えれば良いでしょう。

$$19 \times 25 = (2^4 + 2^1 + 2^0) \times 25$$
$$= 25\times2^4 + 25\times2^1 + 25\times2^0$$
$$= 25\times2\times2\times2\times2 + 25\times2 + 25$$
$$= 400 + 50 + 25$$

二進法のかけ算には，九九は必要ではありません。たとえば 19 × 25 の計算は，

$19 = (10011)_b$, $25 = (11001)_b$ から，

					1	0	0	1	1
×)					1	1	0	0	1
					1	0	0	1	1
		1	0	0	1	1			
	1	0	0	1	1				
	1	1	1	0	1	1	0	1	1

$(111011011)_b = 2^8 + 2^7 + 2^6 + 2^4 + 2^3 + 2^1 + 2^0 = 256+128+64+16+8+2+1 = 475$ のようになります。

実際のコンピュータでどのようにかけ算を実現するかには，いろいろなバリエーションが考えられます。九九と同じように，表を使った計算方法が使われる場合が実際には多いようです。いずれにしても電子回路を使ってかけ算をすることができるわけです。