組み合わせ回路と演算

山本昌志*

2003 年 10 月 27 日

1　初めに

本テキストには、実験実習「基本論理回路 II」を行う上で必要な基礎知識が記述されています。内容は以下の通りです。

- 組み合わせ回路と順序回路
- 加算器
- エンコーダーとデコーダー
- 累算器を用いた演算

2　組み合わせ回路と順序回路とは

皆さんは、2 年生の電子計算機の授業で、ブール代数を学習し組み合わせ回路というものを理解していると思います。もう一度復習すると、論理回路を大別すると、

組み合わせ回路 出力がそのときの入力の状態のみで決まる回路です。これは、出力が以前の動作に依存しないことを言ってます。要するに情報を記憶しない回路です。入力の組み合わせにより出力が決まるので、組み合わせ回路と呼ばれます。

順序回路 出力は、そのときの入力と、その以前の状態で決まる回路です。これは、出力が以前にも依存することを言ってます。以前の状態に依存すると言うことは、以前の情報を記憶していると言うことです。これは入力の順序により、出力が決まるので順序回路と呼ばれます。フリップフロップ回路がこれにあたります。

とに分けられます。今まで、学習してきた回路は組み合わせ回路のみですが、ここでの実験では、順序回路も使われます。ここでの実験の加算器とエンコーダー、デコーダーの回路は組み合わせ回路です。一方、累算器[1]を用いた演算には順序回路が使われています。

*国立秋田工業高等専門学校　電気工学科

[1]通常はアキュームレーターとカタカナで書かれることが多い。演算用のレジスターと考えてよい

ただし、ここでは順序回路を意識する必要は無く、累算器を用いた演算の方法を理解する実験になっています。

組み合わせ回路は、散々学習してきましたので、順序回路についてほんの少し後の方で、かじって見ることにします。

3　1 ビット加算器

論理回路を用いた加算器には、下からの桁上げを考慮しない半加算器 (Half Adder) とそれを考慮した全加算器 (Full Adder) にが有ります。それぞれについて、学習しましょう。

3.1　半加算器

入力 A と B を加算する回路を考えよう。もちろん、それぞれの入力は 0, 1 のいずれかです。1 桁の 2 進数の加算回路です。入出力をブラックボックスで書くと、図 1 のようになります。入力 A と B に対して、出力が S と C です。S は和を表し英語の SUM から、C は桁上がりを表し Carry の頭文字です。このように、入出力だけ示して、中身が分からないものブラックボックスと言います。この 1 ビットの加算のブラックボックスの機能は、

- 加算する各 1 ビットを入力 A と B に入れる。実際の回路では、A と B に対応する端子に 5V を印加[2]する。
- A と B の加算結果の 1 桁目を S から、桁上がりを示す 2 桁目を C から出力する。実際の回路では、S と C に対応する端子に 5V が出力される。

です。これが、1 ビットの加算に必要な全てです。

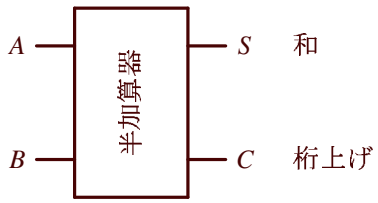

図 1: 半加算器のブラックボックス

さて、中身の回路はどうなっているのでしょうか?。それを考えるには、真理値表を書いてみるのが最も良いでしょう。入出力から、真理値表は表 1 のようになります。

[2]TTL の場合。

表 1: 半加算器の真理値表

A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

残る問題は、この真理値表を実現する回路を設計することだけです。真理値表から、論理回路を求める方法はいろいろ有りますが、この程度で有れば主加法標準形に直すのが簡単でしょう。それぞれは、

$$S = \bar{A} \cdot B + A \cdot \bar{B} \tag{1}$$

$$C = A \cdot B \tag{2}$$

となります。式 (1) の方は排他的論理和 (Exclusive OR) と呼ばれ

$$S = A \oplus B \tag{3}$$

と書かれることも多いです。論理式ができたので、それを回路に直すだけです。これらを表す論理回路は、図 2 や 3 になります。1 桁の半加算器はこれでおしまい。

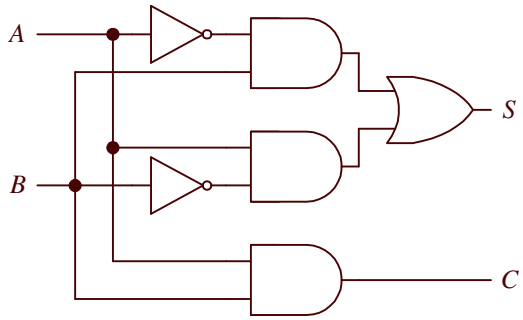

図 2: OR と AND、NOT による半加算器。式 (1) と (2) の回路

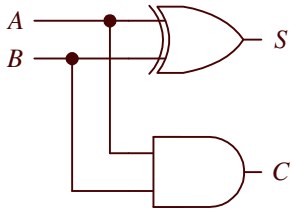

図 3: XOR と AND による半加算器。式 (3) と (2) の回路

3.2 全加算器

先ほどの半加算器は、入力が演算の対象の A と B だけでしたで、1 ビットの加算しかできません。複数のビットの加算を行う場合、下位からの桁上がりも考慮する必要があります。この下位からの桁上がりを考慮した回路が全加算器です。全加算器の入出力をブラックボックスで書くと、図 4 のようになります。入力は 3 個の 1 桁の 2 進数なので、ブラックボックスの入力端子は 3 個になります。一方、出力はその和をあらわし、その最大は 2 桁の $(11)_2$ となります。従って、出力端子は 2 個必要になります。

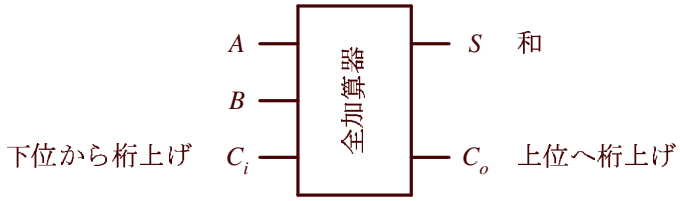

図 4: 全加算器のブラックボックス

先ほどと同じように、真理値表からこのブラックボックス内部の論理回路を考えます。演算の対象が 2 つから 3 つに変わっただけです。3 つの 1 ビットの和を考えればよいのです。その加算の演算の真理値表を表 2 に示します。

表 2: 全加算器の真理値表

A	B	C_i	S	C_o
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

真理値表ができたので、次はこれから論理式を作ります。標準展開を用いて論理式を書くこともできますが、ここではカルノー図を使うのが適当でしょう。この真理値表の出力 S と C_o それぞれのカルノー図は、図 5 と 6 のようになります。これから、論理式を導き出しましょう。論理式は、いろいろな形に展開可能ですが、半加算器の結果の式 (1) や (2)、(3) に近い形に変形することを考えます。これは後で分かるように、全加算器は半加算器と OR ゲートで可能であることを示すためです。

A B \ C_i	0	1
0 0		1
0 1	1	
1 1		1
1 0	1	

図 5: 全加算器の和 S のカルノー図。

A B \ C_i	0	1
0 0		
0 1		1
1 1	1	1
1 0		1

図 6: 全加算器の桁上がり C_o のカルノー図。

最初に述べたように、全加算器は A と B、C_i の和と桁上がりを計算している回路です。従って、A と B、

C_i には区別は全くありません。従って、出来上がった論理式は、それらを入れ替えても成り立つ必要があります。このようなことを考えながら、式の展開を行うと計算が上手になります。

まず初めに和 S の論理式を求めますが、そのとき以下の関係式を使います。

$$\begin{aligned}\overline{\bar{A}\cdot B + A\cdot\bar{B}} &= (\overline{\bar{A}\cdot B})\cdot(\overline{A\cdot\bar{B}}) \\ &= (A+\bar{B})\cdot(\bar{A}+B) \\ &= A\cdot\bar{A} + A\cdot B + \bar{A}\cdot\bar{B} + B\cdot\bar{B} \\ &= A\cdot B + \bar{A}\cdot\bar{B}\end{aligned} \tag{4}$$

これに注意しながら、カルノー図から求められた主加算標準形の S を以下のように変形します。

$$\begin{aligned}S &= \bar{A}\cdot\bar{B}\cdot C_i + \bar{A}\cdot B\cdot\bar{C}_i + A\cdot B\cdot C_i + A\cdot\bar{B}\cdot\bar{C}_i \\ &= (\bar{A}\cdot\bar{B} + A\cdot B)\cdot C_i + (\bar{A}\cdot B + A\cdot\bar{B})\cdot\bar{C}_i \\ &= (\overline{\bar{A}\cdot B + A\cdot\bar{B}})\cdot C_i + +(\bar{A}\cdot B + A\cdot\bar{B})\cdot\bar{C}_i \\ &= (\overline{A\oplus B})\cdot C_i + (A\oplus B)\cdot\bar{C}_i \\ &= (A\oplus B)\oplus C_i\end{aligned} \tag{5}$$

非常にきれいな式が出来上がりました。

つぎに、C_o の論理式を作ります。これもカルノー図から、

$$\begin{aligned}C_o &= A\cdot B + B\cdot C_i + A\cdot C_i \\ &= A\cdot B + (A+\bar{A})\cdot B\cdot C_i + A\cdot(B+\bar{B})\cdot C_i \\ &= A\cdot B + A\cdot B\cdot C_i + \bar{A}\cdot B\cdot C_i + A\cdot B\cdot C_i + A\cdot\bar{B}\cdot C_i \\ &= A\cdot B\cdot(1+C_i+C_i) + \bar{A}\cdot B\cdot C_i + A\cdot\bar{B}\cdot C_i \\ &= A\cdot B + (\bar{A}\cdot B + A\cdot\bar{B})\cdot C_i \\ &= A\cdot B + (A\oplus B)\cdot C_i\end{aligned} \tag{6}$$

となります。

以上で全加算器の論理式が完成したわけですが、もうひとひねりしておきます。それは、半加算器の式 (2) と (3) を用いて、全加算器の式 (5) と (6) を書き直します。ちょっと記号の問題がありますので、半加算器の出力を

$$S_1 = A\oplus B \tag{7}$$

$$C_1 = A\cdot B \tag{8}$$

と置き換えます。すると、

$$S = S_1\oplus C_i \tag{9}$$

$$C = C_1 + S_1\cdot C_i \tag{10}$$

となります。ここで、最後のひねりとして、$S_1\cdot C_i = C_2$ を加えます。すると

$$S = S_1\oplus C_i \tag{11}$$

$$C = C_1 + C_2 \tag{12}$$

となります。これで準備は完了です。

これらの式をゲートで組み立てる前に、最後の式 (11) と (12) から、全加算器は半加算器 2 個と OR ゲートでできることが分かります。即ち、図 7 の通りです。この半加算器と OR ゲートを使った動作は、全加算器として動作することが直ぐに理解できると思います。

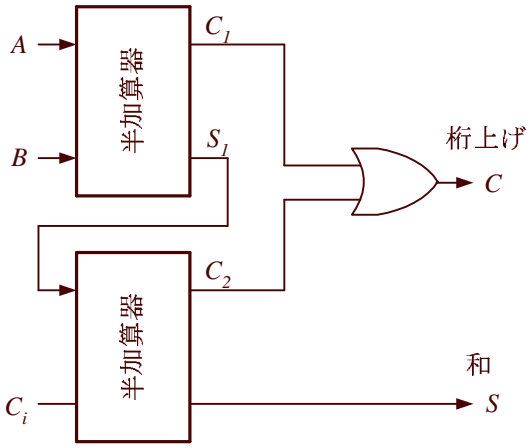

図 7: 2 個の半加算器と OR ゲートで構成される全加算器

全加算器が半加算器と OR ゲートで出来ることが分かったので、論理回路も同じことです。やはり半加算器 2 個と 1 個の OR ゲートで出来ます。図 8 と 9 の通りです。それぞれの論理回路がどの式と対応しているかは分かりますよね。考えてください。全加算器はこれで終わり。

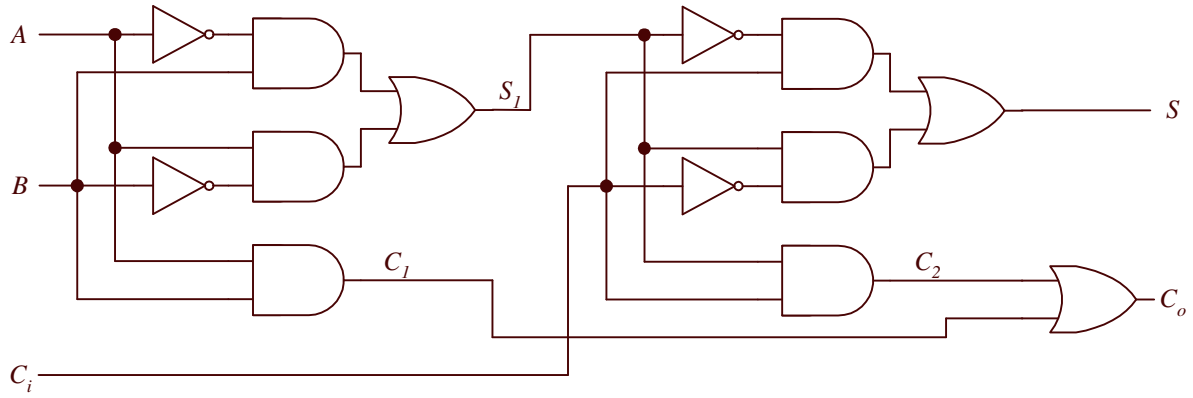

図 8: OR と AND、NOT ゲートによる全加算器

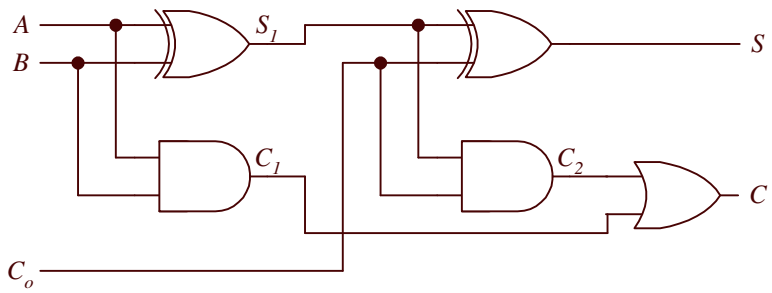

図 9: XOR と AND、OR ゲートによる全加算器

4　エンコーダーとデコーダー

情報を記号で表すことを符号化と言い、その記号を符号といいます。コンピューターでは、記号として 2 進数の (0, 1) を使います。数字や文字、音、絵などを符号化して (0, 1) で表しています。

情報を符号 (コード) 化するための装置をエンコーダー (符号化器) といいます。反対に符号化されたデータを元のデータに戻す装置をデコーダー (解読器) といいます。ここでは、10 進数のデータを BCD コードに変換するエンコーダーと、BCD コードを 10 進数に変換するエンコーダーの実験を行います。

4.1　BCD コード

2 年生の時に学習した BCD コード[3]の復習をします。これは、非常に簡単なコードで 10 進数の 1 桁を 4 桁の 2 進数で表します。2 進数の各桁は重みとして、1、２、４、８を持ちます。すなわち、表 3 のようになります。

この 10 進数と BCD コードの変換する装置 (ハードウェア)、エンコーダーとデコーダーの回路について、次節で学習します。

4.2　エンコーダーとデコーダー

10 進数と BCD を変換するエンコーダーとデコーダーの目的が理解できたと思いますので、具体的な設計に入ります。それぞれの機能をブラックボックスで表すと、図 10 と 11 のようになります。それぞれの入出力は、10 進数の 1 桁を表現するために 10 本の線と BCD コードを表現するための 4 本の線があります。

ここで注意しなくてはならないのは、10 進数を表す線は同時に複数の線が 1 になることは許されないことです。エンコーダーの上流の回路では、それを防ぐ設計が求められます。ここでは、エンコーダーの設計なのでその問題は関係ありませんが、理解しておく必要があります。

そして、デコーダーの 10 進数の線も、入力の 4 本の線に対応してどれか 1 本が 1 になればよいのです。

エンコーダーとデコーダーの入出力がブラックボックスで示されたので、その入出力の関係を表で表します。それぞれ、表 4、5 のようになります。真理値表との違いは 10 進数の部分です。10 進数の部分は、1

[3]Binary Coded Decimals code 日本語では 2 進化 10 進符号

表 3: BCD コード表

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

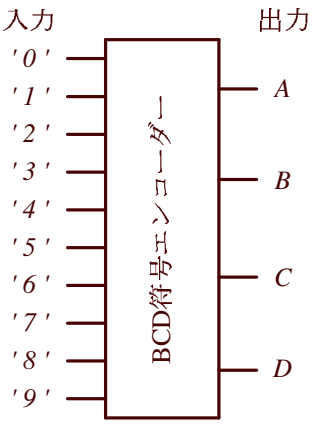

図 10: エンコーダーのブラックボックス

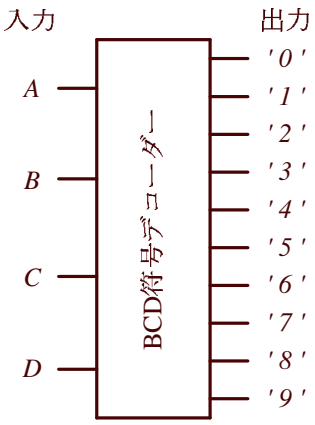

図 11: デコーダーのブラックボックス

となる線を表しています。10 本の線のうちただ 1 本のみが 1 となります。BCD コードと 10 進数との対応の表 3 と同じであることを理解してください。

表 4: エンコーダーの入出力

10 進	BCD 出力			
入力	D	C	B	A
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

表 5: デコーダーの入出力

BCD 出力				10 進
D	C	B	A	出力
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9

エンコーダーとデコーダーの動作が決まったので、残る問題はそれを実現する論理回路を設計することです。表から、エンコーダーの A の線が 1 になるのは、' 1' または'3' または'5' または'7' または'9' の線が 1 の場合です。したがって、それらを 5 入力の OR ゲート入力とし、その出力を A とすればよいのです。B や C、D についても同じです。すると、図 12 のようなエンコーダーの論理回路が書けます。

次はデコーダーです。これは、主加法標準展開で考えるのが簡単です。例えば出力線'1' が 1 になるのは、$A = 1$ かつ $B = 0$ かつ $C = 0$ かつ $D = 0$ の場合です。したがって、$A \cdot \bar{B} \cdot \bar{C} \cdot \bar{D}$ を'1' の出力とすればよいのです。後の線も同じように考えれば、ゲートの配線が決まります。すると、図 13 のようなデコーダーの論理回路が書けます。

5　順序回路の基礎

この部分の用意が出来ませんでした。4 年生で学習します。あるいは、論理回路のフリップフロップを自分で学習してください。

6　N ビットの加算と減算

ここでは、N ビットの加算と減算の回路を考えます。N ビットの加算回路の構成がどのようになっているか良く理解してください。そして、計算が回路で可能であることを体でもって感じ取ってください。

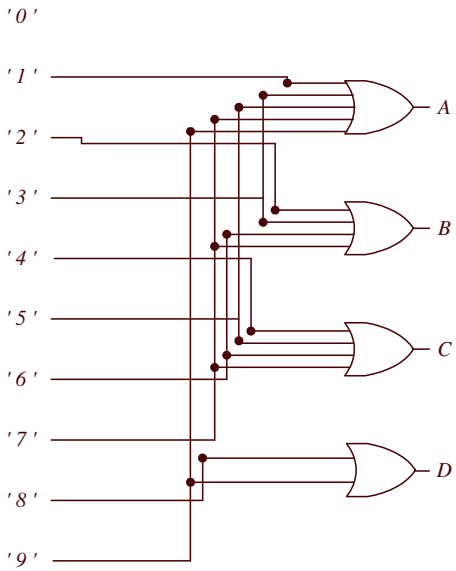

図 12: エンコーダーの論理回路

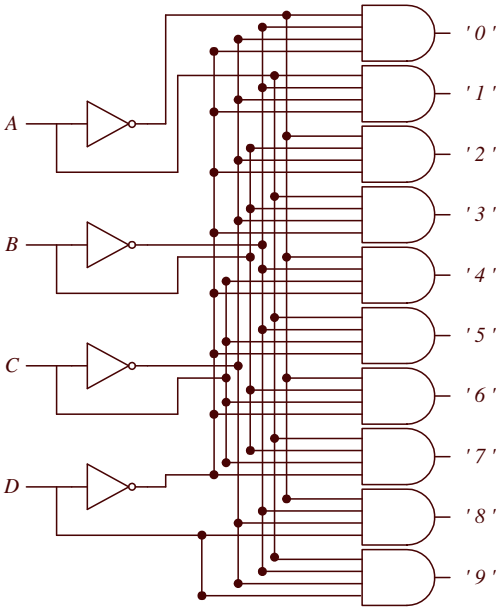

図 13: デコーダーの論理回路

6.1 並列加算回路

並列加算回路の実験は行いませんが、単純なので説明をしておきます。実験する回路が直列加算回路なので、並列加算回路も興味が湧くと思いますので説明の価値はあるはずです。

このプリントの最初の方の「1 ビット加算器」のところで、1 ビットの加算器の回路について説明しました。賢明な諸君であれば、N ビットの加算は N 個の全加算器用いればよいと直ぐに気が付くと思います。即ち、図 14 のようにすればよいのです。これを並列加算器と言います。全加算器は直列に接続されているのですが、加算データは並列に入力されるのでその名前がつけられと思います。

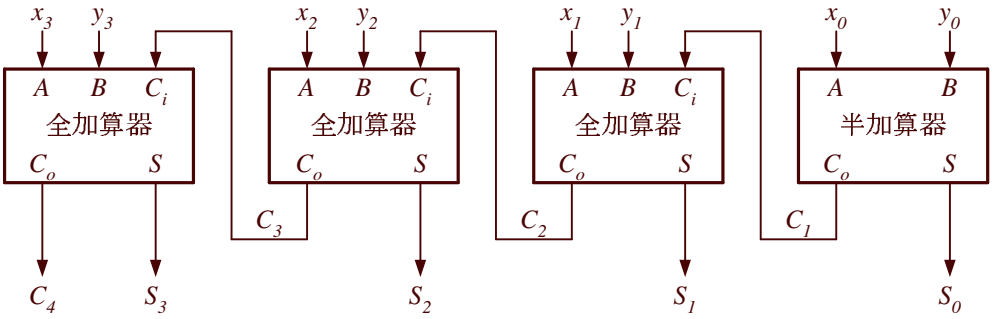

図 14: 4 ビット並列加算器。2 進数の $x_3x_2x_1x_0$ と $y_3y_2y_1y_0$ の加算を行う。

6.2 直列加算回路

これまでの回路は、組み合わせ回路でした。しかし、次の実験では累算器[4]を用い計算を行います。これは、記憶するこでき、順序回路となっています。

直列加算回路は、図 15 に示すように 1 個の全加算器と、2 個のレジスターと 1 個の桁上げメモリーから構成されています。これらの回路の詳細については、時間の都合でここでは述べません。その動作を述べます。

この回路の動作の順序は、次のようになります。

1. 図に示すように計算する数 $x_3x_2x_1x_0$ と $y_3y_2y_1y_0$ を各レジスターに格納します。
2. 次に、各レジスタを 1 ビット右にシフトさせます。あふれ出たビットは、全加算器の入力 A, B となります。最初、C_i は 0 です。したがって、最下位の和が計算され、その結果が S と C_o に出力されます。
3. 計算結果の S は累算器の最上位のビットに格納されます。C_o は桁上げメモリーに格納されます。
4. 次の桁の計算は、先ほどとおなじで各レジスターを 1 ビット右にシフトさせます。あふれ出たビットは、全加算器の入力 A, B となります。同時に、桁上げメモリーから先ほどのデータを引き出し C_i の入力とします。これで次の桁の計算結果が、S, C_o に現れます。
5. 加算すべきビットが全て出て行くまで、以上を繰り返します。
6. 全てのビットの処理が終了したならば、累算器には加算結果が残ります。そして、桁上げメモリーにはオーバーフローのデータが残ります。

以上が、直列加算回路の動作です。

CASL II で SUB GR1,GR2 の動作を思い出してください。GR1+GR2 の演算結果が GR1 に残る、まさにこの回路の動作です。そして、フラグレジスタの OF には最後の C_o の値が設定されます。

6.3 補数器と加算器を使った減算

ここで言う補数とは、2 の補数のことです。前期の授業で学習したように、コンピューター内部では、負の整数は 2 の補数で表します。2 の補数は以下のようにして求めます。

1. 負の数の絶対値を 2 進数で表し、そのビット反転させる。
2. 反転したビットに+1 加算する。

これで、負の数を表します。このように下場合、負の数の加算は加算器で可能です。加算結果も、最上位のビットが 1 ならば、負の数となります。

したがって、減算の演算も減算対象を 2 の補数で表して、加算器で加えることで可能となります。その減算器を図 16 で表します。

[4]アキュームレーター (accumulator) と呼ばれることが多い。演算用のレジスターと考えれば良い。

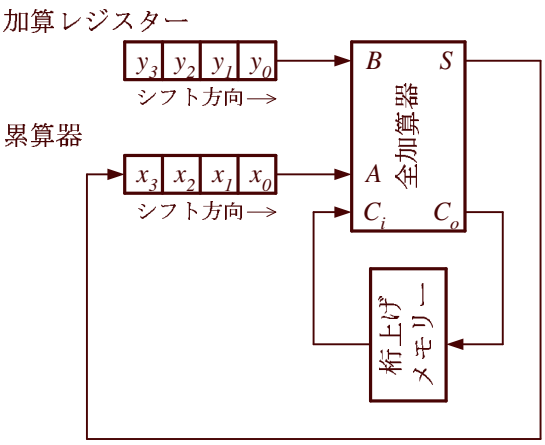

図 15: 4 ビット直列加算器

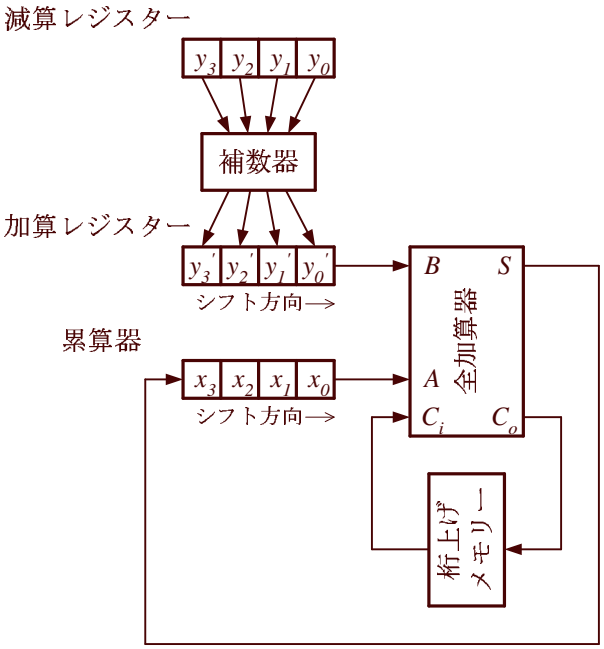

図 16: 補数器と加算器を使った減算器