計算機システムⅠ・第6回

2023年5月24日

今回の内容

6.1 加算器

前回までに、論理素子を組み合わせることで論理関数の計算ができることを学びましたが、この応用として、整数の四則演算など、さまざまな計算を行うこともできます。今回は、二進法で表現された非負の整数の足し算を行う方法について解説します。

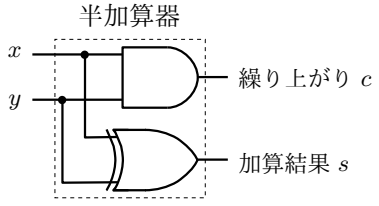
		1	0	0	1
	+	1	0	1	1
(繰り上がり)	1		1	1	
加算結果	1	0	1	0	0

人間が、筆算で何桁かの十進数の足し算を行う際の手順を考えてみると、その基本となっているのは、1桁どうしの足し算です。各位の0～9の数字を足し合わせて、その位の結果を求め、繰り上がりがあればそれを上位の桁で足し合わせるという作業を行っていきます。論理回路で二進数の加算を行う際にも、1桁どうしの二進数(つまり、2つの0/1)の加算が基本となります。与えられた二進数の各位の0/1を足し合わせて、同じ位の結果(0/1)を求め、繰り上がりがあればそれを上位の桁で足し合わせて行きます。たとえば、論理回路で4桁の二進数1001と1011の足し算を行う様子は、右上のような筆算を行うのと同じです。このような原理で二進数の足し算を行う論理回路を作ることができます。

メモ

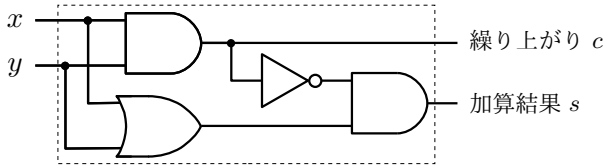
半加算器　二進数の足し算を行う場合に、まず基本となるのは、1桁の二進数を加算する論理回路です。2つの論理変数 x、y の値を加算した結果は下の真理値表のようにならなければなりません。x と y がともに1の場合は、繰り上がりが起こることに注意しましょう。繰り上がりが起こるかどうかは x と y を AND ゲートに入力することで分かります。また、加算結果の同じ位の値は、ちょうど x と y に対する XOR ゲートの出力と同じになっていますので、この論理関数は、下の図のように、AND ゲート1個と、XOR ゲート1個を使って実現することができます。この2入力2出力の論理回路を**半加算器**と呼びます。

入力		出力	
x	y	繰り上がり	加算結果
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0



この半加算器ではXORゲートを使用していますが、XORゲートを使わない場合、例えば、次のような論理回路でも半加算器を実現できます。

半加算器 (別の実現方法)



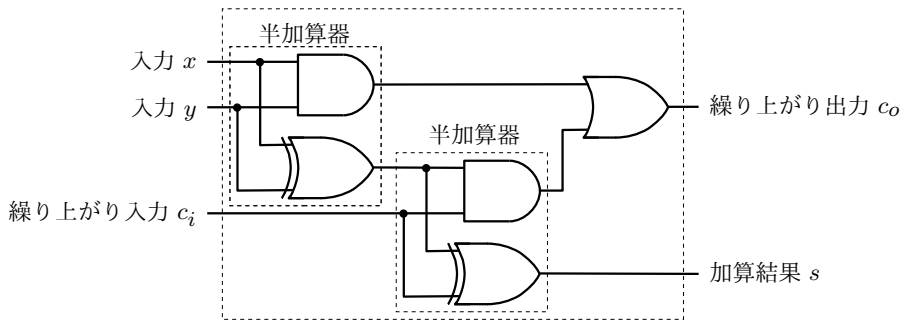
メモ

全加算器　半加算器を使うと、1 桁の二進数の足し算を行って、1 の位の計算結果と、上の位 (2 の位) への繰り上がりを求めることができますが、これだけでは、まだ複数桁の二進数の加算を行うことはできません。下の位からの繰り上がりを、その位に足し込むことが必要です。このため (1 の位を除けば) それぞれの位で、与えられた 2 つの二進数の同じの位の数字 (0/1) と、下の桁からの繰り上がりの、総計 3 つの 0/1 の加算を行わなければならないことになります。

入力			出力	
x	y	繰り上がり	繰り上がり	加算結果
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

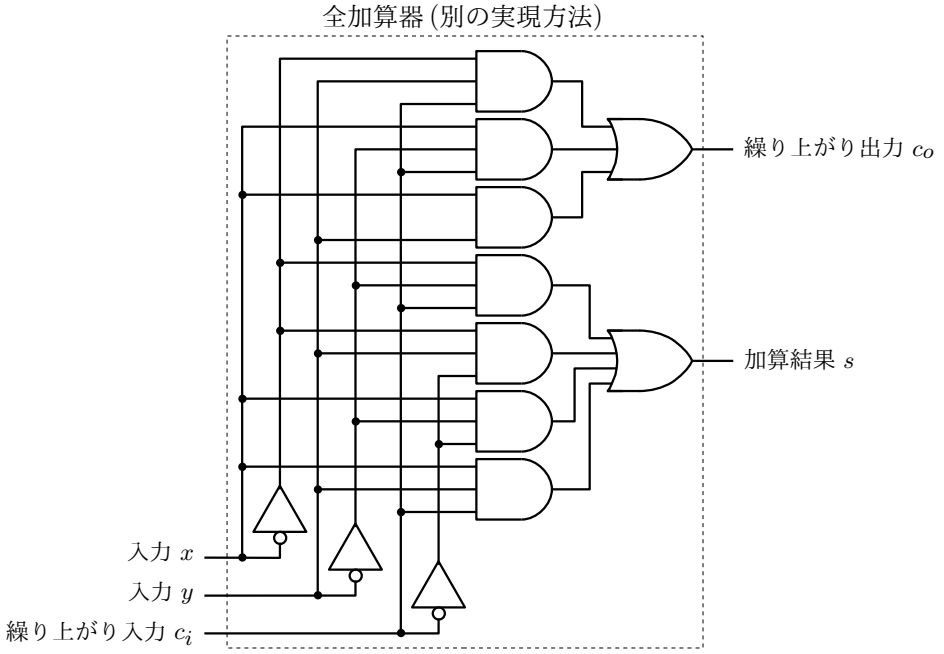
この 3 つの 0/1 の加算の入出力関係は右上の真理値表のようになり、2 つの半加算器を下図のように組み合わせることで実現できます。こうしてできた 3 入力 2 出力の論理回路を**全加算器**と呼びます。

全加算器

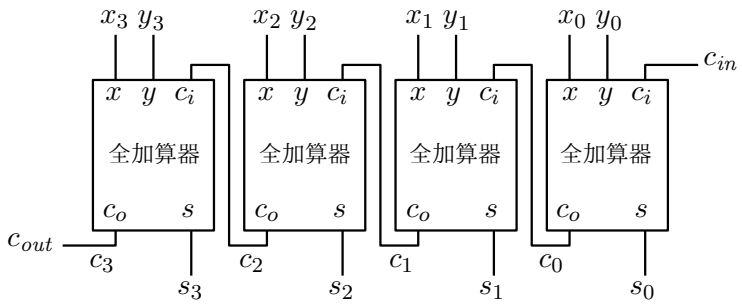


メモ

XOR ゲートを使わないで半加算器を実現すると入力から出力まで最大 3 つのゲートを経由しますから、半加算器を 2 つ並べて全加算器を構成すると最大 6 つのゲートを経由してしまいます。XOR ゲート使わないのであれば、例えば、次のような論理回路にすると、(論理素子の数は多くなりますが) より少ない段数で全加算器を実現できます。



多bit長の二進数の加算器　全加算器を n 個使用することで、n bit の二進数の加算を行うことができます。下の図の論理回路は 4 bit の加算器です[1]。

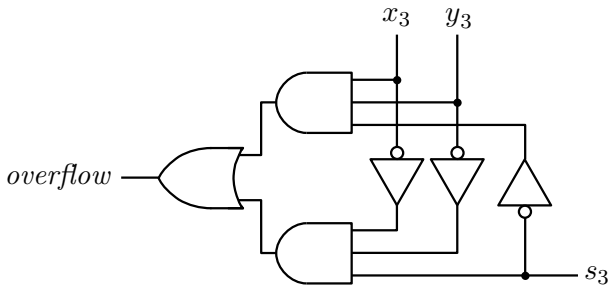


この 4 bit 加算器は、次の筆算に対応する計算を行います。c_{in} や c_{out} は、4 bit を越える長さの二進数をいくつかの 4 bit 長の二進数に分割して計算を行う際や、この加算器を応用して二進数の減算を行う際に使用されます。通常の加算では、c_{in} に 0 を入力します。

		x_3	x_2	x_1	x_0
	$+$	y_3	y_2	y_1	y_0
(繰り上がり)	c_3	c_2	c_1	c_0	c_{in}
加算結果	c_{out}	s_3	s_2	s_1	s_0

[1] このような加算器の場合、x_i や y_i $(i = 0, 1, 2, 3)$ の各入力の値が決まっても、繰り上がりの情報が下位から上位に向って、すべての全加算器を通過してしまうまで計算結果は定まりませんので、bit 長が長くなると、1 回の加算に必要な時間が長くなってしまいます。実際の CPU で採用されている論理回路では、もっと高速に繰り上がりを計算する方法などを使って計算時間を短縮する工夫がなされています。

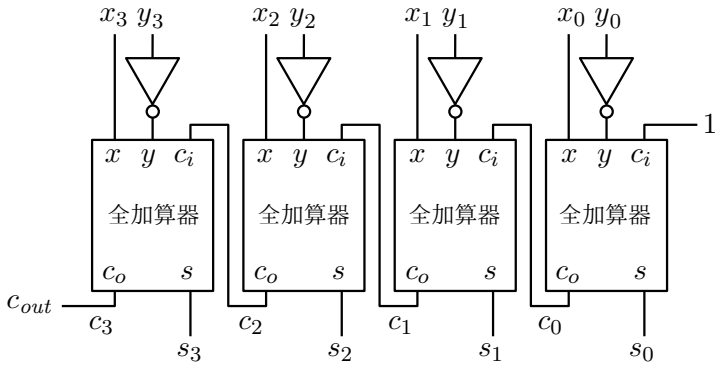
符号付き整数の加算　最上位 bit で符号を区別し、負の整数は 2 の補数で表現した、n bit の符号付き整数の加算も同じ加算器で実現することができます。この加算器でオーバーフロー[2]が起きたかどうかは次のような論理回路 (4 bit 加算器での例) で判定できます。



x_3 と y_3 は入力となった 4 bit の 2 つの符号付き整数の符号 bit に、s_3 はその和 (4 bit) の符号の bit に対応していることに注意してください。

メモ

多bit長の二進数の減算器　n bit の符号付き 2 進表現では、整数 p を表現しているビット列の各ビットを反転し (ビットごとに否定をとり)、1 を加えることで、整数 $-p$ のビット列表現を得ることができます。このため、引かれる数はそのまま、引く数は NOT ゲートを介して加算器に入力し、加算器の c_{in} に 1 を入力することで、減算器を実現することもできます。次は 4 bit の減算器の論理回路です。



計算機システム I ・第 6 回・終わり

[2]オーバーフローとは、n bit の符号付き整数表現の加算で、その n bit の計算結果を見たとき、2 つの正の数の和が負になったり、2 つの負の数の和が正になったように見えることを言います。