2004年ハードウェア構成法中間試験の解説

石井 康雄*

平成17年11月16日

1 比較器・加算器

問題1

4 bitの整数$A(a_3, a_2, a_1, a_0), B(b_3, b_2, b_1, b_0)$がある
$A - B - 1 \geq 0$を出力する回路を求めよ

1.1 入出力ビット数

まず、入出力ビットに関して検証します。入力に関してはAとBの2つの4 bitの整数のみを取り、出力は自明に1 bitの値となります。従って、入出力のビット数に関しては以下のようになります。

入力 $a_3, a_2, a_1, a_0, b_3, b_2, b_1, b_0$

出力 s

従って、この8本の入力ビットを適当に加工して1ビットの結果をもとめればよいということが分かるでしょう。

1.2 解法

この問1の解法は大きく分けて2つの方法があります。それは以下の2つです。

- 符号付整数の加算器
- 正数の比較器

符号付整数の加算の場合には答えが5ビットの値となりますので1ビット符号拡張して普通に加算を行えばよいでしょう。正数の比較器を作る場合には最上位bitを反転させて正数の比較とする必要があります。それぞれ見てみましょう。

*yishii@is.s.u-tokyo.ac.jp

1.2.1 符号付整数の加算

ハードウェアでは直接加算はできるけれど減算はできない…。従って、減算を加算形式に変えましょう。

$$\begin{aligned} A - B - 1 &= A + (\overline{B} + 1) - 1 \\ &= A + \overline{B} \end{aligned}$$

となるのでこれを符号拡張して計算すればよいだけになります。ちょっと工夫をするなら 4 ビットの加算器に少し工夫をするだけで OK になります。

1.2.2 正数の比較

そのままでは正数でも何でもないので多少工夫をします。

$$\begin{aligned} A - B - 1 \geq 0 &\Leftrightarrow A > B \\ &\Leftrightarrow (A + 8) > (B + 8) \end{aligned}$$

一般に n bit の符号付整数は以下のように表せます。

$$A = -2^{n-1} a_{n-1} + \sum_{i=0}^{n-2} 2^i a_i$$

それを符号なし整数にするには定数 (2^{n-1}) を足して a_{n-1} を反転させてしまえばよいわけです。そして、これは今回の問題では両辺に定数 (8) を足すことで実現できるわけです。

2 積和演算

問題 2

6 bit の整数 $A(a_3, a_2, a_1, a_0)$、4 bit の整数 $B(b_3, b_2, b_1, b_0)$、
8 bit の整数 $C(c_7, c_6, c_5, c_4, c_3, c_2, c_1, c_0)$ がある
$A \times B - C - 9$ を出力する回路を求めよ

2.1 入出力ビット数

1 章と同じく入出力のビット数に関して検証します $-32 \leq A \leq 31, -8 \leq B \leq 7, -128 \leq C \leq 127$ ですので簡単な計算で $-384 \leq A \times B - C - 9 \leq 375$ が分かります。従って、この計算の出力は 10 ビットの値を用いれば十分に表せるということが分かります。以上より、入出力のビットをあらわすと以下のようになります。(このときにワラスツリーの出力を計算できます、詳しくは後述)

入力 $a_5, a_4, a_3, a_2, a_1, a_0, b_3, b_2, b_1, b_0, c_7, c_6, c_5, c_4, c_3, c_2, c_1, c_0$

出力 $s_9, s_8, s_7, s_6, s_5, s_4, s_3, s_2, s_1, s_0$

以降ではどのようにこの 10 ビットの出力を得ればよいのかを考えます。

2.2 解法

乗算のアルゴリズムに関しての論述がなかった人が多かったので…。まず整数乗算に関する一般的なアルゴリズムを述べておきます。その後に今回の問題である積和演算の解法に関して考えます。

2.2.1 整数乗算の一般的なアルゴリズム

n bit の整数 A と m bit の整数 B の乗算 AB を求めることを考えます。
A と B のビット列の表す整数値は以下のようになります。

$$A = -2^{n-1}a_{n-1} + \sum_{i=0}^{n-2} 2^i a_i$$

$$B = -2^{m-1}b_{m-1} + \sum_{j=0}^{m-2} 2^j b_j$$

従って、その積 AB は以下のように表せます。

$$\begin{aligned} AB &= 2^{n+m-2}a_{n-1}b_{m-1} + \sum_{i=0}^{n-2}\sum_{j=0}^{m-2} 2^{i+j}a_i b_j \\ &- \sum_{j=0}^{m-2} 2^{n+j-1}a_{n-1}b_j - \sum_{i=0}^{n-2} 2^{n+i-1}a_i b_{m-2} \end{aligned} \tag{1}$$

ただし、この式 (1) では負の整数が多数含まれていて厄介ですのでこれを消去することを考えます。$-a_{n-1}b_j$ などを $(1 - a_{n-1}b_j) - 1$ と置換してみます。これを (1) に代入して整理してみると以下のような式になります。

$$\begin{aligned} AB &= 2^{n+m-2}a_{n-1}b_{m-1} + \sum_{i=0}^{n-2}\sum_{j=0}^{m-2} 2^{i+j}a_i b_j \\ &+ \sum_{j=0}^{m-2} 2^{n+j-1}(1 - a_{n-1}b_j) + \sum_{i=0}^{n-2} 2^{n+i-1}(1 - a_i b_{m-2}) \\ &- \sum_{j=0}^{m-2} 2^{n+j-1} - \sum_{i=0}^{n-2} 2^{n+i-1} \end{aligned} \tag{2}$$

となります。$1 - a_{n-1}b_j = \overline{a_{n-1}b_j}$ と置くことができますのでさらにこの式は以下のように整理できます。

$$\begin{aligned} AB &= 2^{n+m-2}a_{n-1}b_{m-1} + \sum_{i=0}^{n-2}\sum_{j=0}^{m-2} 2^{i+j}a_i b_j \\ &+ \sum_{j=0}^{m-2} 2^{n+j-1}\overline{a_{n-1}b_j} + \sum_{i=0}^{n-2} 2^{n+i-1}\overline{a_i b_{m-2}} \\ &- \sum_{j=0}^{m-2} 2^{n+j-1} - \sum_{i=0}^{n-2} 2^{n+i-1} \end{aligned} \tag{3}$$

ついでに $-\sum_{j=0}^{m-2} 2^{n+j-1} - \sum_{i=0}^{n-2} 2^{n+i-1}$ はただの定数ですから K とでもおいてしまいましょう。最終的に AB の値は以下のように表せます。

$$\begin{aligned} AB &= 2^{n+m-2}a_{n-1}b_{m-1} + \sum_{i=0}^{n-2}\sum_{j=0}^{m-2} 2^{i+j}a_ib_j \\ &+ \sum_{j=0}^{m-2} 2^{n+j-1}\overline{a_{n-1}b_j} + \sum_{i=0}^{n-2} 2^{n+i-1}\overline{a_ib_{m-2}} + K \end{aligned} \tag{4}$$

この式を筆算形式で書くことを考えて見ましょう ($n \geq m$ を仮定します)。

$$\begin{array}{cccccccccc}
 & & & & \overline{a_{n-1}b_0} & a_{n-2}b_0 & & \cdots & & a_0b_0 \\
 & & & \cdots & \cdots & & & & \cdots & \\
 & & \overline{a_{n-1}b_{m-2}} & a_{n-2}b_{m-2} & & \cdots & & a_0b_{m-2} & & \\
 & a_{n-1}b_{m-1} & \overline{a_{n-2}b_{m-1}} & & \cdots & & \overline{a_0b_{m-1}} & & & \\
k_{n+m-1} & k_{n+m-2} & & & \cdots & \cdots & \cdots & & & k_0 \\
\hline
s_{n+m-1} & s_{n+m-2} & & & \cdots & \cdots & \cdots & & & s_0
\end{array}$$

こんな具合になります。そして、通常はワラスツリーを用いて項数を減らしてから加算をします。

2.2.2 今回の課題の場合

乗算をしてから減算を行ってもいいのですがせっかくハードウェアでやるのですべての結果を畳み込みましょう。すると、$A \times B - C - 9$ は筆算形式では以下のようになります。C に関しては $C + 128$ として (つまり最上位 bit を反転させて) 最終的に定数の畳み込みで解決しています。最後の定数に関しては $A = B = C = 0$ のときに $-9(= 1111110111)$ を返すように設定してあります。

$$\begin{array}{cccccccccc}
 & & & & \overline{a_5b_0} & a_4b_0 & a_3b_0 & a_2b_0 & a_1b_0 & a_0b_0 \\
 & & & \overline{a_5b_1} & a_4b_1 & a_3b_1 & a_2b_1 & a_1b_1 & a_0b_1 & \\
 & & \overline{a_5b_2} & a_4b_2 & a_3b_2 & a_2b_2 & a_1b_2 & a_0b_2 & & \\
 & a_5b_3 & \overline{a_4b_3} & \overline{a_3b_3} & \overline{a_2b_3} & \overline{a_1b_3} & \overline{a_0b_3} & & & \\
 & & c_7 & \overline{c_6} & \overline{c_5} & \overline{c_4} & \overline{c_3} & \overline{c_2} & \overline{c_1} & \overline{c_0} \\
0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
\hline
s_9 & s_8 & s_7 & s_6 & s_5 & s_4 & s_3 & s_2 & s_1 & s_0
\end{array}$$

あとは、これを計算する回路をがんばって描くのみです。あとは体力勝負 !! がんばりましょう。

2.2.3 補足:ワラスツリーに関して

ワラスツリーで入力を減らすときには最終型を想像しながら行いましょう。これは非常に簡単なアルゴリズムで考えられます。たとえば今回のケースでは (ありえない話ですが) 上の全ての変数項が 1 であっ

たと仮定 ($a_ib_j = 1, \overline{a_ib_j} = 1, c_i = 1$) します。すると上の筆算形式の演算は以下のように計算できます。

						1	1	1	1	1	1
					1	1	1	1	1	1	
				1	1	1	1	1	1		
			1	1	1	1	1	1			
				1	1	1	1	1	1	1	1
+)	0	1	1	0	1	0	0	0	0	0	
	1	1	1	1	1	1	1	1	1	1	
+)	1	0	0	1	0	1	0	0	0	1	
	FA	HA	HA	FA	HA	FA	HA	HA	HA	HA	

このときにはワラスツリーを用いて 10 本の出力と 4 本の出力を得ることができます。各ビットに関して 1 が 2 つたっているところは最終的にフルアダーをもちいて、1 が 1 つしかたっていないところは最終的にハーフアダーで最終結果を計算することになりますですから、上の筆算式での FA はフルアダー、HA はハーフアダーを示しています。

また、この式の結果はこのように馬鹿正直に筆算をする必要はなく、概念さえ分かっていれば変域の計算の際に求めることができます。今回は $-384 \leq A \times B - C - 9 \leq 375$ が分かっていますから答えは高々10 ビットです。

次に A、B、C が全て正数であったと仮定をしましょう。また、畳み込む定数も正数化します (0110100000 = 416 ですね)。また、10 ビットの正数の最大値は 1023(= 1111111111) であることに注意すると。

$$A \times B - C - 9 = 63 \times 15 + 255 + 416 = 1616 = 1023 + 593 = 1111111111 + 1001010001$$

となることが分かります。この計算で上の筆算での最終結果の 2 つの数字と同等の結果が得られます。これによりワラスツリーの出力結果を比較的簡単に得ることができます。ワラスツリーは結構複雑になりがちですからこのようにして最終の入力ビットの本数をあらかじめ計算しておくとケアレスミスを避けることができます。