2 × 2 部分乗算器を用いた高速並列乗算器の構成

非会員 李 相球 (韓南大学)

正員 秋月 影雄 (早稲田大学)

Fast Parallel Multiplier using 2×2 Submultipliers

Sang-Gu Lee (Hannam Univ.)

Kageo Akizuki (Waseda Univ.)

In this paper, we propose a novel parallel multiplication scheme using 2×2 submultipliers. It adopts a new encoding method which halves the number of partial products through 2×2 submultipliers and rearrangement of primitive partial products. We present the parallel multiplication algorithm and design a 16bit×16bit multiplier architecture of the proposed method, which is suitable for VLSI implementation. The proposed scheme can be easily extended to large-size multipliers with 32, 54 or even longer bits. This parallel multiplication scheme can be applied to many areas such as DSP, MPEG and so on.

キーワード: parallel multiplication, 2×2 submultiplier, partial product

1. はしがき

並列乗算はデジタル計算機のみならずデジタル信号処理における一つの重要な機能である。並列乗算器はアレイ乗算器とツリー乗算器に大別される。基本的にアレイ乗算器は2次元の加算アレイから構成され、その加算アレイは Baugh と Wooley[1]によって提案されたアルゴリズムに基づいている。これに対してツリー乗算器は加算ツリーから構成されている。それらは Wallace[2]によって提案され、Dadda[3]らによって改善された。

これまで研究されてきたほとんどの高速乗算器は Booth エンコーディング[4, 5]と Wallace ツリーに基づいている。一般に、並列乗算は部分積の計算と部分積の和の計算という2段階の演算から成り立っている。Booth エンコーディング手法は部分積の計算に用いることができ、Wallace ツリーは和の計算に用いることができる。Booth エンコーディング手法を用いることによって、部分積を通常の AND 演算に比べて 1/2 に減少することができる。しかし Booth エンコーディング方式は乗数

の変換および部分積の選択のために Booth エンコーダと Booth セレクタが必要で VLSI 実装には不適な信号伝搬回路が必要となる欠点があり[9]、この信号伝搬回路なしに構成することが望まれる。

本研究で提案する並列乗算ブロックは 2×2 の部分乗算器をもち、Booth エンコーディングをもつブロックと同じ部分積を生成するものであるが、信号伝搬回路がないために VLSI 実装にはるかに向いていると言うことができる。

ここでは VLSI 実装に適した 2×2 部分乗算器を用いて並列乗算アルゴリズムとアーキテクチャを示し、16bit×16bit 乗算器を設計し、SPICE シミュレーションを通して評価を行う。

2. 4進数乗算についての定理

デジタル計算機で用いられている乗算器では、それぞれの部分積は被乗数が 1bit か 2bit (Booth's encoding) ずつ左側にシフトされる。被乗数と乗数がそれぞれ 4進数で表現されている場合には、各部分積は 4進数 (0～3) × 4進数

(0～3)の組み合わせであらわれ、その値は0, 1, 2, 3, 4, 6, 9だけの値をもつ。この部分積の各 bit についての次の定理を得ることができる。

[定理]
a,b,c,d,e を4進数の数と仮定する。
 $a \times b, c \times d, 4 \times a \times d, 4 \times b \times e$ の値を2進数 (0, 1) で表わし、それらを各 bit について合計した結果は高々3である。

[構成手法]

block3 の2つの数 a,b を block1 と block2 に別々に置き、block4 からひとつだけ (たとえば、d) を選んで block1 に置く。block2 の2つ目の部分 (たとえば、e) は如何なる数でも良い。このようにすると前述の定理を満足する。

[証明]
上記の4つの積を2進数で表わしそれを各 bit について表わすと次のようになる。

ここで各 bit での和が最大となる位置は 2^3 の位と 2^2 の位である。2つの4進数 (2bit の2進数) を乗算した結果は {0, 1, 2, 3, 4, 6, 9} のいずれかであるから、 2^3 の位で和が4になる条件は次のようである。

『 $a \times d, b \times e$ が各々 2 or 3 or 6 の値をもち、
 $a \times b, c \times d$ は全て 9 である。』
しかし、 $a \times b, c \times d$ が全て 9 になるのは a,b,c,d がすべて3の場合であり、 $a \times d$ が9となるから和は4にならない。また、 2^2 の位で和が4になる条件は $a \times d, b \times e$ が全て奇数かつ $a \times b, c \times d$ が4又は6の場合に限られる。従って、a と b のいずれかが2となるから、 $a \times d, b \times e$ が同時に奇数になることはなく和は4になり得ない。

上記の定理を利用すると4つの1bit を加えるときにも“1”の個数が0～3しか現われないから1つの CSA(Carry Saved Adder)で充分である。

3. 2 × 2 部分乗算器を利用した乗算

一般に n bit の2つの数 A と B の乗算は $n=2m$ とおくと次のように表わすことができる。ここで $A(2m-1:0)$ は数 A が 2m bit で 2^{2m-1} の位から 2^0 の位までの bit 構成であることを示している。

$$\begin{aligned} & A(2m-1:0) \times B(2m-1:0) \\ &= A(m-1:0) \times B(m-1:0) \\ &+ A(2m-1:m) \times B(m-1:0) \times 2^m \\ &+ A(m-1:0) \times B(2m-1:m) \times 2^m \\ &+ A(2m-1:m) \times B(2m-1:m) \times 2^{2m} \end{aligned}$$

従って、n-bit 乗算器は4つの m-bit の部分乗算器で構成することができる。ここでは最も簡単な 2-bit の基本的な部分乗算器を以下のように構成して n-bit 乗算器を構成する。

$$Z(3:0) = X(1:0) \times Y(1:0)$$

$$Z_0 = X_0 \cdot Y_0$$

$$Z_1 = \text{ExOR}(X_1 \cdot Y_0, X_0 \cdot Y_1)$$

$$Z_2 = X_1 \cdot Y_1 \cdot (X_0 \cdot Y_0)$$

$$Z_3 = X_1 \cdot Y_1 \cdot X_0 \cdot Y_0$$

ここで、() は否定を表す。
図1は2×2部分乗算器の論理回路である。ここで $X_{2i+1}X_{2i}$ と $Y_{2j+1}Y_{2j}$ は2×2部分乗算器の入力で、 Z_{ij} は出力である。(i,j=0,1,2,...,m) たとえば、i,j が0のとき、 X_1X_0 と Y_1Y_0 の積は $Z_{00}(3:0)$ の4-bit である。

図1 2 × 2 部分乗算器の論理回路
Fig.1 Logical circuit of 2×2 submultiplier

この2×2部分乗算器の回路図とレイアウトは図2と図3のようである。

図2 2×2部分乗算器の回路図
Fig.2 Circuit diagram of 2×2 submultiplier

図3 2×2部分乗算器のレイアウト
Fig.3 Layout of 2×2 submultiplier

このような2-bitの部分乗算器を利用した8-bitの乗算器は図4のようである。図4でグループ1と2の各ブロックは部分乗算器を表わしている。

図4 8-bit乗算器
Fig.4 8-bit multiplier

ここで、16個の部分乗算器の積の項が生成され、これらは2つのグループに分けることができる。各bitについてグループ1では和が最大となるのが3で、グループ2では4である。しかしここでグループ1と2の各項を前の定理を満足するように適当に交換すると、すべての和の最大は3になる。部分積の項をこのように配置変えたあとの形は図5のようになる。図5でグループ2の各々の項が前の定理を満足することは $B \times P, C \times O, C \times P, D \times O$ の場合は $C \Rightarrow a, P \Rightarrow b, D \Rightarrow c, O \Rightarrow d, B \Rightarrow e$ のように代入すれば直ちに理解できる。また、 $C \times P, D \times O, C \times Q, D \times P$ の場合は $P \Rightarrow a, D \Rightarrow b, C \Rightarrow c, Q \Rightarrow d, O \Rightarrow e$ のようである。

図5 8-bit乗算器(部分積の項の配置変え)
Fig.5 8-bit multiplier
(after rearrangement of partial products)

ここでグループ1はCSAで実現できる。グループ2でも各bitでの和は3以下になる。しかし、この場合4つのbitのいずれかが1か0かわからない。このことを考慮してグループ2に対する回路を設計する。必要な回路は4つの入力を受け入れ、和とキャリーを生成する。普通の場合は2つの全加算器が必要だが、グループ2では入力が4つあっても、それらがすべて1になることはない。ここで、Karnaugh mapを利用して回路を設計する。

a	b	c	d	S	C
0	0	0	0	0	0
0	0	0	1	1	0
0	0	1	0	1	0
0	0	1	1	0	1
0	1	0	0	1	0
0	1	0	1	0	1
0	1	1	0	0	1
0	1	1	1	1	1
1	0	0	0	1	0
1	0	0	1	0	1
1	0	1	0	0	1
1	0	1	1	1	1
1	1	0	0	0	1
1	1	0	1	1	1
1	1	1	0	1	1
1	1	1	1	X	X

$S = a \oplus b \oplus c \oplus d$
 $C = a \cdot d + b \cdot d + a \cdot c + b \cdot c + a \cdot b + c \cdot d$

ここで a,b,c,d は図 5 のグループ 2 のための 4 つの入力で、S は和、C はキャリーである。グループ 2 は前の定理を満足するので、a,b,c,d がすべて 1 になることはなく、a,b,c,d がすべて 1 の場合、S と C は X (don't care 条件) になる。1 つの CSA は全加算器と同じである。グループ 2 に対する回路は全加算器のゲートレベルと同じ段数を持っているので、グループ 1 の CSA で要する遅延時間と同じである。このような 4:2 Compressor (4-input CSA)の回路図とレイアウトは図 6 と図 7 のようである。

図 6 4:2 Compressor の回路図
Fig.6 Circuit diagram of 4:2 Compressor

図 7 4:2 Compressor のレイアウト
fig.7 Layout of 4:2 Compressor

4. 16 bit 乗算器の構成

16bit の場合、2 × 2 部分乗算器を利用し、図 8 のように構成する。ここでは 4 つのグループが

図 8 16-bit 乗算器
Fig.8 16-bit multiplier

生成され、グループ 2, 3, 4 はすべて前の定理を満足する。

n-bit の乗算器では (n/2)² の部分積が生成される。これらの中、overlap されない部分積を連結すると (n-1)個の部分積ワード (submultiplication word) が生じ、これらを 1 つの 3-word group (3つの部分積ワード) と (n/4-1)個の 4-word group (4つの部分積ワード) に結ぶことができる。ここで 3-word group の定義は Group1 での乗算結果 A × O, A × Q, E × O…による系列、B × O, A × R, F × O…による系列と A × P, B × Q, C × R…による系列の 3-word をもつ group である。また 4-word group は 4 個の系列をもつ group である。この時、4-word group 内の部分積らをほかの 4-word group 或いは 3-word group 内の部分積と適当に交換すると 4-word group 内の隣接するすべての部分積の対を前の定理を満足するように配置変えることができる。

	3-word group の項数	4-word group の項数
8-bit	10	6
16-bit	22	22, 14, 6
32-bit	46	54, 46, 38, 30, 22, 14, 6

一般に、3-word group の項数は $(n/2-1) \times 3+1$ である。また、4-word group の項数は $n/4-1$ である。n-bit の時、4-word group の各々の項数は $8i-2$ ($i=1,2,\dots,n/4-1$) である。任意の bit の場合にも前の定理を活用できるように 4-word group を構成できる。8-bit,16-bit の場合は図 5、図 8 で示した。また、16-bit 以上の場合はコンピュータシミュレーションで構成できることを確認した。しかし、8,16bit を除いた任意の bit の場合、提案した定理が使用できるように並べかえるためのシステムティックな方法は見つかっていない。上記に述べた部分ブロックと Wallace ツリーを利用して設計した 16bit×16bit 全体乗算器のブロック図は図 9 のようである。

図 9 16-bit 全体乗算器のブロック図
Fig.9 Block diagram of 16-bit multiplier

本論文で提案する方式と Booth エンコーディング方式の比較のために 4bit×6bit の例を図 10 で示す。ここで PP0,PP1,PP2 は部分積である。部分積の数は同じだが、Booth エンコーディング方式では Booth エンコーダと Booth セレクタが必要であるが、提案する方式では 2×2 部分乗算器、4-input CSA で構成できる。

Booth エンコーディング方式との比較のため、同じ工程を利用して設計した。使用した技術は VTI 社の $0.5 \mu\text{m}$ 標準 CMOS 工程である。(1-poly 2-metal; metal width $1.4 \mu\text{m}$) 同じ Wallace ツリーを使用するので、パイプラインステージのためのラッチは構成しないで入力から Wallace ツリー入力

図 10 Booth エンコーディング方式と提案する方式の比較のための 4b×6b の例
Fig.10 Example of 4b×6b encoder using Booth's and the proposed scheme

までの遅れを検討した。提案された乗算器のエンコーディングブロック (2×2 部分乗算器) が外部入力を直接受け入れる反面 Booth セレクタはエンコーダから入力を受け入れるので、理論的にはエンコーダのトランジスタの大きさはセレクタのトランジスタの大きさの 16 倍になるはずであるが、設計では少しの遅れを容認して適当な大きさにし、2つの乗算器の全体の大きさが同じになるように調整した。図 11 と図 12 は各々 Booth エンコーダと Booth セレクタを示している。2つの乗算器の全体面積は Booth エンコーダが $3300 \mu\text{m} \times 1200 \mu\text{m}$ で、提案されたのは $2600 \mu\text{m} \times 1500 \mu\text{m}$ である。

図 11 Booth エンコーダ
Fig.11 Booth's encoder

図 12 Booth セレクタ
Fig.12 Booth's selector

図 13 SPICE シミュレーション結果
Fig.13 Results of SPICE simulation

図 14 乗算器の全体レイアウト
Fig.14 Total layout of multiplier

レイアウト上での寄生要素を考慮して回路を抽出して SPICE シミュレーションした結果は図 13 で、提案された方法を利用した乗算器の全体レイアウトは図 14 である。Booth エンコーダおよび提案したエンコーダをブリチャージしたあと、入力 (Y-input) を印加してから (上の graph) 部分積が生成されるまでの遅れを比較した。Booth エンコーダの場合、部分積を生成するまでの遅れは $t(\text{Booth}) = t(\text{Booth-encoder}) + t(\text{Booth-selector})$ で、提案されたエンコーダの場合の遅れは $t(\text{proposed}) = t(2 \times 2 \text{ submultiplier}) + t(4\text{-input CSA})$ である。図 13 で Booth encoder の出力 (中央の graph, 実線) は sign bit をふくめて 17 個の Booth selector 入力を駆動するので立ち上がり時間が長い (0.8ns)。一方、提案した 2×2 部分乗算器 (下の graph, 実線) は外部入力を印加したとき、ただ 1 つの 4-input CSA 入力を駆動するだけなので立ち上がり時間が短くなり遅れも 0.5ns ぐらいである。また、2 番目の stage、すなわち Booth selector の出力 (中央の graph, 点線) と 4-input CSA の出力 (下の graph, 点線) は各々 0.46ns、0.54ns で大きい差がないことがわかる。入力が入ってから出力 (Wallace ツリーへの入力) が出る時まで Booth エンコーダとセレクタはすべて 1.26ns に対し、提案された方式はすべて 1.04ns で、0.22ns 程度早いことがわかる。

これまで報告されている CMOS 乗算器と本論文で提案した乗算器の性能を比較すると表 1 のようである。表 1 で既存の乗算器はその発表データをそのまま示したものである。本論文で実際に SPICE シミュレーションした結果は、チャンネル長 $0.8 \mu\text{m}$ で遅れが 5.4ns である。Yano の論文では遅れがこれより短くなっているが、そこでは既存の CMOS ではない新しい CPL (Complementary pass-transistor logic) 技法でできた CPL full adder を使っており、その 1 つの CPL full adder の遅れ時間は 0.26ns と報告されている [4]。従って、本論文で

表1 乗算器らの性能比較
Table 1 Performance comparison of multipliers

	ワード長 (bit)	チャンネル長 (μ m)	遅れ (ns)	面積 (mm ²)
Hanawa[6]	54	0.3	4.3	16.96
Hilker[7]	53	0.8	7.5	9.0
Kernhof[8]	16	2.0	39.2	4.6
Yano[4]	16	0.5	3.8	4.03
Proposed	16	0.8	5.4	17

提案した乗算器に Yano の手法の Wallace ツリーと CLA を用いたとすると次式により遅れ時間は 3.64ns と推定される。

$$1.04 + 10 \times 2.6 = 3.64$$

ここで、1.04 は本論文で提案した方法による部分積生成までの遅れで、10 は full adder chain の段数である。

この結果から本論文で提案した部分積生成法は良い結果を与えるものと言える。32bit 以上の乗算器ではこの速度の差がもっと著しいと考えられる。また、提案したエンコーディング手法は被乗数と乗数が互いに対称的であり、外部入力が入力ブロック内のブロックを直接駆動するので Booth エンコーディングの場合の一方の入力に対する遅れの問題がなく、全体的に高速になる。

5. むすび

本論文では 2×2 部分乗算器と 4 進数での定理を利用して新しいエンコーディング方式を提案した。提案された方式は、乗算で生ずる規則を利用してハードウェア面積をへらし、演算速度を向上することができる。本研究で新しく発見した 4 進数乗算に対する定理を利用して Booth エンコーディング方式のような部分積を持つ新たな方式を提案し、設計した。提案された方式は被乗数と乗数に対して対称的で、Booth エンコーディング方式のような信号伝搬回路なしに構成できる。

0.5μ m 標準で乗算器を設計し、SPICE シミュレーションで Booth エンコーディング方式に対し同一の面積でより高速であることを確認した。同じ原理を適用して 24-bit とか 54-bit 以上の乗算器を制作することができ、この高速並列乗算器は幅広く使用できる。今後の課題としては 2 の補数も考慮して提案した定理を満足する乗算器の構成方法についての研究がある。

(平成 9 年 10 月 23 日受付, 同 10 年 9 月 25 日再受付)

文献

[1] C. R. Baugh and B. A. Wooley, "A Two's Complement Parallel Array Multiplication

Algorithm," IEEE Trans. On Computers, Vol. 22, No. 12, pp. 1045-1047, Dec. 1973.

- [2] C. S. Wallace, "A Suggestion for a Fast Multiplier," IEEE Trans. On Electronic Computers, pp. 14-17, Feb. 1964.
- [3] L. Dadda, "Some Scheme for Parallel Multipliers," Alta Frequenza, Vol. 19, pp. 349-356, May. 1965. Reprinted in Computer Design Development, E. E. Swartzlander Jr., Ed., Hayden Book, Rochelle Park, New Jersey. 1976.
- [4] K. Yano, Y. Yamanaka, T. Nishida, M. Saito, K. Shiomhigashi, and A. Shimizu, "3.8-ns CMOS 16X16 multiplier using complementary pass-transistor logic," IEEE JSSC, vol. 25, no. 2, Apr. 1990, pp. 388-395.
- [5] M. Uya, K. Kaneko and J. Yasui, "A CMOS floating point multiplier," IEEE Int. SSCC Conf., Digest of Technical Papers, Feb. 1984, pp. 90-91.
- [6] M. Hanawa, K. Kaneko, T. Kawashimo and H. Marayama, "A 4.3 us 0.3 μ m CMOS 54 x 54b Multiplier using Precharged Pass-transistor Logic," IEEE Int. Solid State Circuits Conf., Digest of Technical Papers, 1996, pp. 364-365.
- [7] S. Hilker, N. Phan and D. Rainey, "A 3.4 us 0.8 μ m BICMOS 53b x 53b Multiplier Tree," IEEE Int. Solid State circuits Conf., Digest of Technical Papers, 1994, pp. 292-293.
- [8] J. Kernhof, M. Selzer, M. A. Beunder and et al., "Mixed Static and Domino Logic on the CMOS Gate Forest," IEEE JSSC, vol. 25, no. 2, Apr. 1990, pp. 396-402.
- [9] S. Y. Kung, "On Supercomputing with Systolic/Wavefront Array Processors," Proceedings of the IEEE, Vol. 72, No. 7, pp. 867-884.

李 相球 (非会員)

韓国のソウル大学電子工学科、韓国科学技術院を卒業し、1983年韓南大学の教授となり、現在に至る。コンピュータ構造、ファジィ理論、並列処理の研究に従事。

秋月 影雄 (正員)

1965年3月早稲田大学大学院工学研究科電気工学専攻博士課程修了。1963年同大学助手、専任講師を経て、1972年同教授となり、現在に至る。ストカスティックシステム理論、信号処理の研究に従事。工学博士。IEEE、電気学会、計測自動制御学会などの会員。