超高精度整数乗算器の高速化とモジュール化

正 員 中村 次男 (国際短期大学)

非会員 佐藤 友威 (東京電機大学)

非会員 鈴川 敦之 (東京電機大学)

非会員 阿部 一広 (東京電機大学)

非会員 山口 芳男 (東京電機大学)

正 員 笠原 宏 (東京電機大学)

Speeding up and Modularizing the Ultra-high Precision Integer Multiplier.

Tsugio Nakamura, Member(Kokusai Junior College)

Tomotake Sato, Non-member, Nobuyuki Suzukawa, Non-member,

Kazuhiro Abe, Non-member, Yoshio Yamaguchi, Non-member,

Hiroshi Kasahara, Member(Tokyo Denki University)

With the expectancy of higher performance of digital systems, the multiplier with both higher speed and higher precision is required as a key factor of the system improvement. On the other hand, it is quite common that these digital parts are assembled as a system LSI, and in this case, it is essential that these parts are re-usable as a core. The answer to this requirement is *modularizing*, which can cope with the variable precision, not a fixed one, as a multiplier structure.

The structure proposed here can perform a multiplication by cascading necessary number of the same modules and by repeated usage of the modules with necessary clock cycles. By this method, much higher speeding up compared to the conventional sequential type, as much as limitless multiplication precision compared to the parallel type, can be obtained.

キーワード：高速乗算，可変長乗算，拡張機能，チップスライス，モジュール化

1. まえがき

ディジタルシステムにおける基本演算機能である乗算の使用頻度は高く、更なる高速化が望まれているが、それとともに益々高精度化も要求され、これに対応して演算器の集積度やファンイン ファンアウトおよび開発コストなどの増加が問題となる。超高精度乗算や可変精度乗算時の効率的な実行などを実用面から考慮すると、順序回路形乗算機構が適していると言える。以上の諸問題はハードウェアによるチップスライス化（モジュール化）によって解決できる。本論文で提案する乗算器はモジュールごとの乗算を並列化して高速化を図り、必要な回数の繰り返し演算によ

って乗算結果を得る方式である。完全な並列形乗算器に対して入出力数とゲート数が少なく済み、従来の順序回路形乗算器に対しては、高精度になるほど大幅な高速化が期待できる。本方式を用いれば 1 モジュールの乗算精度を 16 ビット精度とした場合、順序回路形に対して 18 倍、32 ビット精度とした場合約 38 倍の高速化が実現できる。

乗算についてはこれまでさまざまな研究がなされ、いろいろな乗算器が提案されてきた^{(1)~(4)}。乗算は加算の繰り返し演算のため、加算時の桁上げ伝搬遅延時間をいかに小さくするかが重要となる。これには桁上げ先見法や冗長 2 進法を用いた乗算機構がよく知られている^{(3) (4)}。

一方、集積回路技術の著しい進歩によるディジタルシス

テムの高精度化の例としては、例えば、情報セキュリティ技術として不可欠な RSA 公開鍵暗号システムが挙げられ、現在では 1,024 ビット以上のべき乗剰余演算が行われている。このような超高精度化に対しては乗算器の入出力数とゲート数増が大きな問題となる。また、数百万ゲート規模のシステム LSI が次世代の半導体生産として注目されている。このような大規模な集積回路が可能になると全ての回路ブロックを最初から設計することは困難であり、開発元の異なるマクロセル（コア）の再利用が容易なアーキテクチャであることが求められている⁽⁵⁾。

このようなことから、多精度およびコアの再利用可能性と高速性を考慮するとハードウェアをモジュール化することは今後益々重要になってくるといえる。モジュール化が実現できれば、小規模化によって入出力数とゲート数増を抑制でき、演算に必要なモジュール数の単純なカスケード接続だけでダイナミックな任意精度演算が可能となる。このようなモジュールはコアの再利用性を考慮したアーキテクチャに適している。従来では集積化完了後の精度増要求に対しての対処法は困難であったが、同一モジュール数の追加で対応でき、しかも小規模化での集積化などによる、低コストで、高速な乗算機構が構築できることになる。

本論文ではまず、これまで報告されてきた高速乗算器について説明し、次に提案する乗算器について述べ、最後に他方式との性能比較評価を行う。

2. 従来の高速乗算器

デジタルシステムの高精度化に伴い、重要な基本演算のひとつである乗算には高精度で、かつ高速化がデジタルシステムの性能向上の要因として求められている。乗算の高速化の手法としては、1 サイクルに乗数を 2 ビット以上処理する多数ビット走査手法と更に 2 の補数表示数を直接乗算できる Booth の方法、また、符号付きディジット (SD) コードの冗長性を利用した乗数リコード法などが知られている⁽¹⁾。これら順序回路形乗算器を更に高速化するため、組合せ回路形による並列乗算器や桁上げ先見方式、SD 表現の一種で基数 2 の冗長 2 進表現を用いた乗算器などが提案されている。

本章では最も高速乗算向きアーキテクチャとして知られている冗長 2 進表現を用いた乗算器と通常の 2 進表現による並列形について考察する

＜2.1＞ 冗長 2 進表現を用いた乗算器 基数が 2 の SD 表現の一種である冗長 2 進表現では各桁を {−1,0,1} の要素で表し、一桁を 2 ビット必要とする。この冗長 2 進表現を用いると並列加算は演算桁数に関係なく一定時間で行うことができる。n ビット乗算には冗長 2 進表現の n ビットの部分積を n 個、加算することにより積を求める。最後に冗長 2 進表現で得られた積を通常の 2 進数に変換する。部分積の加算に冗長 2 進加算木を用いて 2 つずつ木状に加え合わせると $\log_2 n$ 段の加算で行え、演算時間は $O(\log_2 n)$ という

報告がある⁽³⁾。

最近の報告では一桁の表現に 2 ビットと 3 ビットを混合させた構成で更に高速化を図り、64 ビット精度で乗算時間が 58.48ns、ゲート数が 121,160 という乗算器が報告されている⁽⁴⁾。

＜2.2＞ 並列形乗算器 n ビット並列乗算器は n^2 個の AND ゲートと、n ビットの並列加算器を 1 ビットシフトした状態で $(n-1)$ 段構成することにより実現できる。4 ビット並列乗算器の構成図を図 1 に示す。n ビット並列乗算器のゲート数は $11n^2-9n$ 、最大遅延時間は NAND, NOR, NOT の基本ゲートの遅延時間を 1 [△] とすると $4n$ [△] になる。

図 1 4 ビット並列乗算器の構成図

Fig.1 Structure of 4-bit parallel multiplier.

以上の代表的な高速アーキテクチャを考察する。2.1 の冗長 2 進表現を用いた方法は演算桁数に関係なく一定時間で演算を実行でき、最も高速な手法であることが最大の利点であり、高精度になるほど顕著になる。しかし、一桁を 2 ビット以上で表現するため、バス幅、ゲート数および入出力数が増大する。2.2 の並列形は冗長 2 進表現による手法よりはバス幅、ゲート数および入出力数が 1/2 以下で済むが、精度増に伴い、並列加算器の伝搬遅延が大きくなる。

いずれも、ある精度内での高速化を実現するものであり、ワンチップ化後の精度を超えた使用、超高精度演算およびコアの再利用性といったアーキテクチャに適しているとは言い難い。

3. 提案する乗算器アーキテクチャ

集積回路技術の向上により、デジタルシステムは益々高機能、高精度化が進むものと予測される。精度増に対する演算器の再開発に要するコスト、様々なデータの演算に対し、その精度に適した演算器による効率的な演算実行およびコアの再利用性可能なアーキテクチャなどを考慮した乗算器のチップスライス化を提案する。

現在、最も高速な手法として知られている 2. で述べた冗長 2 進表現による乗算器はその回路の複雑さからチップスライス化を考慮した報告はない。並列形乗算器は筆者らが先に開発してある⁽⁶⁾が、1 チップの演算精度の k 倍には k^2 個のチップを必要とし、RSA 公開鍵暗号のべき乗剰余演算に求められる 1,024 ビット以上というような超高精度には実用上問題となる。そこで、チップスライス化した順序回路形乗除算器^{(7) (8)}によってゲート数および入出力数を抑えて RSA 公開鍵暗号システムに内蔵して試作を行った⁽⁹⁾。

しかし、順序回路形はクロックを用いる分、並列形に比べると演算速度が低下する。そこで、1 チップ (モジュール) 内を並列形で高速化を図り、その繰り返し演算によって乗算結果を得る方式を提案する。チップスライス化を考慮した本乗算器は並列形に比べて、ゲート数が約 1/2 と少なく済み、順序回路形よりも大幅な高速演算が期待できる。

〈3. 1〉本乗算器の概要 説明を簡単にするため、1 モジュール 4 ビット乗算器の例を図 2 に示す。単純な同一モジュールのカスケード接続だけで、演算精度に合ったモジュール数により効率的な乗算が行えるようなアーキテクチャとするため、以下の機能を従来の並列形乗算器に組み込む。

- (1) カスケード接続用演算データ伝搬の入出力
 - (2) 繰り返し演算機構
- カスケード接続用入力は下位モジュールからの部分積加算結果の桁上げ入力 4 ビット ($CI_3 \sim CI_0$) と上位モジュール

(a) 1モジュールのブロック図

(b) 4ビット×1ビット乗算／4ビット並列加算器の概要

図 2 4 ビット乗算器モジュール
Fig.2 4-bit multiplier module.

ルからの部分積下位 3 ビットの被加数入力 ($Z_2 \sim Z_0$) で、カスケード接続用出力は部分積加算結果の桁上げ出力 3 ビット ($CO_2 \sim CO_0$) から成る。並列加算器 4 段目の桁上げ出力 (桁上げデータの最上位ビット) は精度拡張時、上位モジュールへの桁上げとして伝搬し、最上位モジュールでは自モジュール内に被加数としてフィードバックされる。

繰り返し演算機構は各モジュールの部分積と新部分積の加算用に 4 ビットの入力 $L_3 \sim L_0$ を設ける。このとき、次の乗数との乗算結果である新部分積が生じるまで、旧部分積を保持しておく必要があるため、 $L_3 \sim L_0$ のデータをラッチする機構とする。その信号が部分積ラッチ信号入力 LUT である。カスケード接続により精度を拡張した乗算において、最初の下位モジュール分の乗数との乗算時は旧部分積がないので、ラッチ出力を加算しないようにするために部分積の切換信号 SW を設けてある。

〈3.2〉モジュール化による精度拡張 n ビット乗算器モジュールの k 倍精度の乗算には k 個の同モジュールをカスケード接続するだけで乗算機構が構成できる。乗算操作は各乗算モジュールに被乗数を分散して与えた状態で、先ず最下位モジュール分の乗数 n ビットを両モジュールに与え ($n=4$ では $B_3 \sim B_0$)、全被乗数との部分積を得てラッチする。このとき最下位モジュールから積の最下位 n ビットが出力されている ($n=4$ では最下位モジュールの $M_3 \sim M_0$ から積の P_3

$\sim P_0$)。次にその上位分の乗数 n ビットを両モジュールに与え ($n=4$ では $B_7 \sim B_4$)、この部分積とラッチしておいた部分積との加算結果をラッチする。このときも積の最下位 n ビットの次の上位 n ビットが最下位モジュールから出力される ($n=4$ では同様に最下位モジュールの $M_3 \sim M_0$ から積の $P_7 \sim P_4$)。

以上の操作を順次、最上位乗数 n ビット分まで行う。乗算終了時は残りの積が下位 n ビットから順に、最下位モジュールの下位 n ビット ($n=4$ では最下位モジュールの $M_3 \sim M_0$ から積の $P_{kn-1} \sim P_{kn-4}$)、その上位の積 n ビットは最下位モジュール上位 ($n-1$) ビットとその上位モジュールの ($n+1$) ビット目から出力され ($n=4$ では最下位モジュールの $M_6 \sim M_4$ から積の $P_{kn+2} \sim P_{kn}$ 、その上位モジュールの M_3 から P_{kn+3} が)、順次出力されて、最上位モジュールの上位 n ビットから積の最上位 n ビット ($n=4$ では最上位モジュールの $M_7 \sim M_4$ から積の $P_{2kn-1} \sim P_{2kn-4}$) が出力される。このときの各部分積のラッチ回数は $(k-1)$ 回になる。図 2 の 4 ビット乗算モジュールを 2 モジュールカスケード接続して 8 ビット乗算を行う例を図 3 に、そのタイミングチャートを図 4 に示す。被乗数の上位 4 ビット $A_7 \sim A_4$ を上位のモジュール 2 に、そして、下位 4 ビット $A_3 \sim A_0$ を下位モジュール 1 の被乗数としてそれぞれ与えた状態で、先ず、下位の乗数 4 ビット $B_3 \sim B_0$ を両モジュールの乗数として与える。その全被乗数と乗数下位 4 ビットとの部分積をラッチ機構にラッチする。この

図 3 2 倍精度乗算の例 (8 ビット $\times$ 8 ビット)

Fig.3. Example of double precision multiplication. (8-bit $\times$ 8-bit).

図 4 2 倍精度乗算のタイミングチャート (8 ビット×8 ビット)

Fig.4. Timing chart of the multiplier expanded by two modules.
(Example of the one multiplier module for 4 bits)

図 5 乗算過程と部分積のデータ伝搬

Fig.5. Multiplication process and the propagation of its partial product.

ときモジュール 1 の $M_3 \sim M_0$ に積の最下位 4 ビット $P_3 \sim P_0$ が出力されている。次に乗数の上位 4 ビット $B_7 \sim B_4$ を両モジュールに与え、その部分積とラッチしておいた部分積との加算を行って残りの乗算結果を得る。従って、部分積切替機構はこのとき、ラッチ機構の出力側に切り替わっている。

図 5 は以上の 2 モジュールに渡る乗算過程と部分積のデータ伝搬を示したものである。図では全被乗数 $A_7 \sim A_0$ の 8 ビットと乗数下位 $B_3 \sim B_0$ の 4 ビットとの乗算過程を示したもので、LUT 信号でこの部分積をラッチする。このとき、部分積の切替機構は図のようにグランド側となっており、ゼロ

の部分積との加算を実行したことになる。次に、乗数を上位の $B_7 \sim B_4$ に代える。そして、切替機構を SW 信号により部分積のラッチ側 (同図、部分積の切替機構のスイッチが右側) に切り替えて、同様に $A_7 \sim A_0$ と $B_7 \sim B_4$ との乗算結果とラッチしておいた部分積との加算によって、最終結果を得る。

4. 性能評価

提案する乗算器はチップスライス化によって任意精度の乗算を効率的に演算可能であることを大きな特徴とするものである。冗長 2 進表現を用いた手法では高精度での入出力

数とゲート数増を伴い、チップスライスは容易ではなく、筆者らの知る限り、これまで実現には至っていない。そこで、筆者らが先に開発した並列形⁽⁶⁾と順序回路形⁽⁷⁾に対する乗算器の性能評価を行う。

ゲート数と遅延時間は2入力 NAND, NOR および NOT の基本ゲートを1ゲート、1[△]として⁽¹⁾、nビット乗算器について算出し、次に、乗算精度拡張時について、比較評価を試みる。

＜4.1＞ゲート数 乗算精度nビットの並列乗算器は図1で示すように、nビットの並列加算器を1ビットシフトした状態で(n-1)段構成することにより実現できる。しかし、チップスライスを考慮した場合、精度拡張時は他チップ(モジュール)から部分積が伝搬する。そのため、nビット並列加算器はn段構成となる。1チップ精度のk倍精度には同一チップを2次元的に配列した構成になりk²個のチップを必要とする。全加算器は9ゲートで構成できる。

nビット並列形乗算器のゲート数

$$9 \times n \text{ 並列加算器} \times n \text{ 段} + 2 \text{ AND} \times n^2 \text{ 個} = 11n^2 \quad \text{[ゲート]}$$

順序回路形は並列加算器を1段だけ用意し、乗数に比例したクロックを用いて繰り返し演算を行って乗算結果を得る。従って、制御回路が必要となり、nビット乗算器では $15\log_2 n + 72$ ゲートになる。

nビット順序回路形乗算器のゲート数

$$12n \text{ 被乗数ラッチと乗算機構} + 19n \text{ 並列加算器ラッチ機構} + 13n \text{ 乗数ラッチとシフト機構} + (15\log_2 n + 72) \text{ 制御機構} \\ = 44n + 15\log_2 n + 72 \quad \text{[ゲート]}$$

提案するnビット乗算器1チップのブロック図は図2に示すように、nビット精度ではnビット並列形の乗算機構にnビットのラッチと部分積切換機構を付加したものである。

提案するnビット乗算器のゲート数

$$11n^2 \text{ 並列乗算} + 6n \text{ ラッチ} + 2n \text{ スイッチ} + \text{約 } 500 \text{ 制御機構} \\ = 11n^2 + 8n + 500 \quad \text{[ゲート]}$$

＜4.2＞端子数 乗算データの入出力にはバス形式ではなく、従って入出力は多重化せず、独立して並列に入出力する場合の入出力端子数を1チップnビット精度乗算器とした場合で算出する。ただし、電源ピンは含まない。

nビット並列形乗算器

$$n \text{ 被乗数} + n \text{ 乗数} + (2n-1) \text{ 積出力} + 2n \text{ 桁上げ} + (2n-1) \text{ 部分積入力} \\ = 8n-2 \quad \text{[ピン]}$$

nビット順序回路形乗算器

$$n \text{ 被乗数} + n \text{ 乗数} + 2n \text{ 積} + 14 \text{ 制御信号} \\ = 4n + 14 \quad \text{[ピン]}$$

提案するnビット乗算器

$$n \text{ 被乗数} + n \text{ 乗数} + n \text{ 桁上げ入力} + (n-1) \text{ 桁上げ出力} + (n-1) \text{ 被加数入力} + 2n \text{ 積} \\ + n \text{ ラッチ} + 2 \text{ 制御信号} \\ = 8n \quad \text{[ピン]}$$

＜4.3＞伝搬遅延時間 ANDゲートと全加算器の伝搬遅

延時間は2[△]で、最長伝搬遅延を算出する。配線による遅延は実装時、ゲートの配置後に決まるもので、またプロセスによっても異なるため、ここでは含めない。

nビット並列形乗算器

$$2 \text{ AND} + (n+n) \text{ 個} \times 2 \text{ 全加算器} \\ = 4n + 2 \quad \text{[△]}$$

順序回路形は1ビット×nビット乗算機構をn回繰り返し積を得る。その繰り返しごとに部分積のラッチとシフト操作が行われる。この時間的余裕をtとすると以下の関係式で表される。

$$\{(2+n \times 2) \text{ 並列加算器} + 2t \text{ ラッチとシフト}\} \times n \text{ クロック}$$

ラッチとシフトの時間的余裕tは小さすぎると演算誤りの原因になるため、十分な余裕をとる必要がある。最適値に関してはチップ内配線遅延やプロセスの種類など、いろいろ議論されるところであるが、実用に際しての一例として、並列加算器の遅延の10%として算出する。

nビット順序回路形乗算器

$$\{(2+n \times 2) + 2(2+n \times 2) \times 0.1\} \times n \\ = 2.4n^2 + 2.4n \quad \text{[△]}$$

提案するnビット乗算器では1チップ内精度での乗算と繰り返し演算のため、並列形と同じである。

1チップnビット精度での比較を表1に示す。実用に際してはデータの入出力にバスを介して行われるため、端子数はどの方式も16ビットあるいは32ビットといった入出力バスの端子数と制御用信号となる。従って、64または84ピンのパッケージに集積化が可能である。

次に、nビット乗算のk倍の乗算精度に対して考察する。並列形はk²個のチップを2次元的に配列した構成になり、伝搬遅延はk倍になる。

並列形乗算器

$$2 + (nk \text{ 倍} + nk \text{ 倍}) \times 2 \\ = 4nk + 2 \quad \text{[△]}$$

順序回路形はk個のカスケード接続でn×kクロックを要する。

順序回路形乗算器

$$\{(2+2n) + 2(2+2n) \times 0.1 \text{ 時間的余裕 } 10\%\} \times k \text{ 個} \times nk \text{ クロック} \\ = 2.4nk^2(n+1) \quad \text{[△]}$$

提案する乗算器はk個のカスケード接続で(k-1)回の繰り返し演算になる。

提案する乗算器

$$\{2 + (nk \text{ 個} + n) \times 2 + \{2 + (nk + n) \times 2\} \times 0.1\} \times (k-1) \text{ 回} \\ = 2.2(nk + n + 1)(k-1) \quad \text{[△]}$$

1チップの乗算精度を16ビットおよび32ビットとした場合、128ビットおよび1,024ビット精度に対する比較を表2に示す。並列形はクロックを用いず、素子の遅延だけで演算時間が決まり、最も高速であるが、精度拡張時は1チップ精度の倍数の2乗に比例するチップを必要とするため、現実には大きな問題となる。順序回路形は入出力数とゲート数が大幅に少なく済むが、1ステップごとにラッチとシフト操作を繰り返すため、多くの演算時間を要する。

表 1 1チップ内精度乗算の場合の比較

Table 1. Comparison against one chip multiplier precision.

1チップ の精度	回 路 方 式								
	並 列 形			順 序 回 路 形			本 乗 算 方 式		
	ゲート数	端子数	伝搬遅延〔△〕	ゲート数	端子数	伝搬遅延〔△〕	ゲート数	端子数	伝搬遅延〔△〕
16	2,816	126	66	838	78	652.8	3,444	128	66
32	11,264	254	130	1,557	142	2,534.4	12,010	256	130

表 2 精度拡張時のチップ数と伝搬遅延時間

Table 2. Number of chips and propagation delay time at the expanded multiplier.

	1チップ の精度	回 路 方 式					
		並 列 形		順 序 回 路 形		本 乗 算 方 式	
		チップ数	伝搬遅延時間〔△〕	チップ数	伝搬遅延時間〔△〕	チップ数	伝搬遅延時間〔△〕
128ビット	16	64	514	8	41,779.2	8	2,233
精度に拡張	32	16	514	4	40,550.4	4	1,062.6
1,024ビット	16	4,096	4,098	64	2,673,868.8	64	144,282.6
精度に拡張	32	1,024	4,098	32	2,595,225.6	32	72,087.4

＜4.4＞実装面積の一例 VLSI 評価対象としては以上の他に面積や消費電力も含まれる。しかし、これらはプロセスの種類、線幅および配置配線などによって大きく異なる。そこで、筆者らはFPGAでの試作実験を行ったので、FPGAに実装した場合を参考例として表3に示す。表3は1モジュールが4ビット乗算器で、8ビット精度に拡張した場合のFPGA（XILINX, XC5210）内ロジックセルの使用量とその割合を示す。

表 3 FPGA 内ロジックセル数の比較

Fig.3. Number of Logic Cells in FPGA

	乗 算 精 度	
	4ビット(1モジュール)	8ビットに拡張
並列形	90(6.9%)	392(30.2%)
順序回路形	32(2.5%)	64(4.9%)
本乗算方式	99(7.6%)	191(14.7%)

XC5210の総ロジックセル数は1296で、

()内はその割合を示す

提案する乗算器は、精度拡張時は順序回路形と同様に必要倍数個の単純なカスケード接続だけで済む。また、1チップの精度を大きくするほど並列性を増して高速化が図れ、順序回路形に対して、提案する乗算器1モジュールの精度を16ビットとした場合18倍、32ビット精度乗算器とした場合は約38倍の高速化が実現でき、大幅な高速演算が可能である。

5. むすび

デジタルシステムの高精度化が進んでおり、基本演算のひとつである乗算には高精度での高速演算がシステムの性能向上の要因として求められてきている。乗算機構の開発後の仕様変更や演算精度増の要求およびシステムLSIにおけるコアの再利用などを考慮すれば、乗算機構としての演算精

度は固定ではなく、可変精度に対応容易なチップスライス化を可能としたモジュール化は今後益々重要になってくるといえる。

高速手法としては冗長2進表現が挙げられるが、高精度化には入出力数、ゲート数およびバス幅増などの問題に加え、チップスライス化が困難であることから、並列形、順序回路形に対する性能評価を行った。並列形は最も高速であるが精度増に対しては1チップの倍数の2乗に比例したチップを必要とするため、超高精度には現実的ではない。提案する乗算機構は精度拡張に際しては単純な乗算器モジュールのカスケード接続と繰り返し演算により乗算結果が得られ、順序回路形に対し、大幅な高速化が可能になった。これは1モジュールの乗算精度を多くするほど高速化が図れる。

(平成12年6月29日受付、平成13年3月7日再受付)

文 献

- (1) Kai Hwang: “Computer Arithmetic-PRINCIPLES, ARCHITECTURE, AND DESIGN”, John Wiley & Sons, Inc(1979)
- (2) Donald E. Knuth: “The Art of Computer Programming II”, Addison-Wesley(1981)
- (3) 高木直史, 安浦寛人, 矢島脩三: 「冗長2進加算木を用いたVLSI向き高速乗算器」, 信学論, VolJ66-D, No.6, pp.683-690(1983-6)
- (4) 恒川佳隆, 日野杉充希, 斉藤正人, 蛇川勝巳, 三浦守: 「1桁2ビット/3ビット混合型高性能冗長2進加算器とその乗算器への応用」, 電学論C, Vol.119-C, No.5, pp.644-653(1999-5)
- (5) Mark Brinbaum: “How VSIA Answers the SOC Dilemma”, IEEE, COMPUTER, pp.42-50(1999-6)
- (6) 中村次男, 笠原宏: 「拡張機能を持った並列乗算器モジュール」, 電気学会全大, 406(1991-4)

- (7) 中村次男, 笠原宏:「拡張容易な乗算器モジュールとそのワンチップ化の提案」, 電学論 C, Vol.110-C, No.2, pp.95-100(1990-2)
- (8) 中村次男, 笠原宏:「任意の精度に拡張容易な除算器の提案」, 電学論 C, Vol.111-C, No.3, pp.123-128(1991-3)
- (9) 中村次男, 大石博朗, 笠原宏:「RSA 公開鍵暗号システム実装におけるビットスライス化の一方式」, 電学論 C, Vol.118-C, No.7/8, pp.1073-1081(1998-7/8)

中村 次男 (正員) 昭 46 東京電機大学工学部電気工学科卒。平 2 同大学大学院修士課程了。昭 46 日本電気精器株式会社入社。現在, 国際短期大学情報ネットワーク学科助教授。博士(工学)。デジタル演算器, 高速暗号, オブジェクト指向ハードウェア設計法に関する研究に従事。著書「デジタル回路の基礎」(日本理工出版会)他。IEEE 会員。

佐藤 友威 (非会員) 平 12 東京電機大学工学部電気工学科卒。現在, 同大学大学院修士課程在学中。デジタル演算器, 公開鍵暗号処理の高速化に関する研究に従事。電子情報通信学会会員

鈴川 敦之 (非会員) 平 12 東京電機大学工学部電気工学科卒。現在, 同大学大学院修士課程在学中。デジタル演算器, 高速暗号化, オブジェクト指向ハードウェア設計法に関する研究に従事。電子情報通信学会会員

阿部 一広 (非会員) 平 11 東京電機大学工学部電気工学科卒。現在, 同大学大学院修士課程在学中。デジタル演算器, 高速暗号化に関する研究に従事。電子情報通信学会会員

山口 芳男 (非会員) 平 11 東京電機大学工学部電気工学科卒。現在, 同大学大学院修士課程在学中。デジタル演算器, 高速暗号化に関する研究に従事。電子情報通信学会会員

笠原 宏 (正員) 昭 45 東京電機大学博士課程満期退学。同大学助手, 講師, 助教授を経て工学部電気工学科教授。現在, 情報環境学部情報環境工学科教授。工学博士。パワーエレクトロニクス, 計算機システム, 制御用分散処理システム, オブジェクト指向システム設計法, インタラクティブにとらえた脈診情報処理方式, 高速暗号化に関する研究に従事。情報処理学会, 電子情報通信学会, IEEE, ACM 各会員。