社団法人 情報処理学会 研究報告
IPSJ SIG Technical Reports

2005－ＳＬＤＭ－119（２）
2005／3／17

動画像処理向け高速公開鍵暗号LSIアーキテクチャ

金 成男† 土井 伸洋† 田中 初一†† 今井 繁規††† 木村 晋二†

† 早稲田大学 情報生産システム研究科
†† Sharp 株式会社
††† 神戸大学 工学部

あらまし 電波やネットワークを利用したマルチメディアコンテンツの安全な流通のために，安全で高速な公開鍵暗号システムが必要とされている．本研究ではRSAと同じ原理に基づき，復号化処理が二回の乗算と一回の加算からなる新しい暗号系に対して，これを実現する高速なLSIアーキテクチャの提案および設計と評価を行なった．まず，復号化に必要な二回の乗算と一回の加算をまとめ上げ，1つの乗算ループで処理するようにした．また，本暗号では1024bitの長さの剰余演算を用いているので，桁上げのない加算を用いて全体の遅延を小さくしている．さらに，桁上げ無し加算とその計算結果を通常の二進数へ変換するための後処理をパイプライン化すると同時に，複数段の桁上げ無し加算を1クロックで実行することで処理のクロック数を削減している．これらの機構で実時間性能を達成できた．

キーワード 公開鍵暗号，LSIアーキテクチャ，動画像処理

Efficient Hardware Architecture of a New Simple Public-Key Cryptosystem for Real-Time Data Processing

Chengnan JIN†, Nobuhiro DOI†, Hatsukazu TANAKA††, Shigeki IMAI†††, and Shinji KIMURA†

† Graduate School of Information, Production and Systems, Waseda University
†† SHARP corporation
††† Kobe University

Abstract This paper proposes an efficient LSI architecture for deciphering of a new simple public-key cryptosystem to obtain real-time performance in motion picture processing. In the new cryptosystem, the deciphering process just consists of two multiplication and one addition, and these operations are merged into one multiplication operation in our design. The cryptosystem uses long bit integers such as 1024bit, so adders with no carry propagation are adopted. Multiplication is implemented as the repetition of the addition without carry propagation and the result is post-processed to obtain the usual binary numbers. To reduce the total clock cycles, a pipeline architecture for the multiplication and the post-processing is introduced, and a loop-unrolling for the multiplication is applied. With these mechanisms, our LSI can obtain the real-time performance with 81MHz clock.

Key words Public-key cryptosystem, LSI architecture, motion picture processing

1. はじめに

電波やネットワークを利用したマルチメディアコンテンツの流通が盛んになるなかで，セキュリティの確保は最も重要な課題である．セキュリティーを保つために，現在ではRSA暗号に代表される公開鍵暗号[1]が広く使われている．これらの暗号は非常に大きな数の素因数分解が困難であることを利用しており，さまざまな攻撃方法に対し強固な安全性を誇っている．しかしその強固さ故に復号化に必要な計算量は多く，リアルタイムのデータ処理には適用が困難であった．

本稿ではRSA暗号と同じ原理に基づく新しい暗号系[2]について，これを実現する高速なLSIアーキテクチャの提案および設計，評価について述べる．この暗号系においては，復号操作が二回の剰余乗算と一回の加算のみで構成されているため，高速な復号化が可能である．そのため従来適用の難しかった動画像データにも応用することができる．LSIの設計においては，2種の演算を効率良く実現できるよう演算回路を工夫した．

まず剰余乗算の実現方法として，P. Montgomeryによって考案されたモンゴメリ乗算[3]～[6]を利用した．これは剰余乗算を効率良く行なうアルゴリズムのひとつである．剰余乗算では

多ビット整数の加算を繰り返す必要があり，キャリーの伝搬による遅延時間が問題となる．この問題を解決するために，本稿では Carry Save Adder (桁上保存加算) [7]～[9] および Redundant Binary Adder (冗長二進加算) [10] を利用した．これらの加算方式はキャリーの伝搬なしで演算を行なうことができるため，回路の遅延時間はビット長に独立であり，多ビット整数の加算に好ましい．また複数回の加算を 1 段にまとめても遅延はそれほど増加しない．ただし，これらの演算方式によって得られる結果は冗長な表現形式であるため，後処理により通常の二進表現に変換する必要がある．

我々は，剰余乗算における繰返し加算のまとめあげ，剰余乗算と後処理のパイプライン化，二度の乗算を 1 つのループで実現する手法に基づく動画像処理向けの復号化回路を設計し，640×480 pixel - フルカラー (24bit) - 30 フレーム/sec の動画像をリアルタイムに処理できることを確認した．また，本回路を少し変更して二つ用いることで暗号化にも適用できることを示した．

2. 新しい公開鍵暗号系について

ここでは文献 [2] で提案された新しい公開鍵暗号系について述べる．この暗号系は RSA 暗号と同じく素因数分解の難しさを利用した暗号系で，RSA 暗号と同等の強度を備えながら復号化に必要な演算量が少ないという特徴を持つ．

2.1 鍵の生成

まず鍵の生成について述べる．始めにビット長 512bit の素数 p, q を求める．p, q を掛け合わせた数を M とし，法 M の乗法群における最大生成元 g を一つ選ぶ．ここで最大生成元 g は次の条件を満たす法 M 上の数とする．

（1） $0 < g < M$

（2） g は p, q のいずれに対しても素

（3） $g^p (\mathrm{mod}\ M) \neq g$ かつ $g^q (\mathrm{mod}\ M) \neq g$

鍵は 2 つの乱数 s, t (ここではいずれも 512bit とする) より生成する．s, t はそれぞれ $gcd(s, q-1), gcd(t, p-1)$ を満たす数である (gcd は最大公約数)．これをもとに，鍵生成に必要な数 g_1, g_2 を次のようにして求める．

$$g_1 = g^{s(p-1)} \ (\mathrm{mod}\ M) \tag{1}$$

$$g_2 = g^{t(q-1)} \ (\mathrm{mod}\ M) \tag{2}$$

そして $\{M, g_1, g_2\}$ を公開鍵として公開し，$\{p, q\}$ を秘密鍵として保持する．

2.2 暗号化および復号化

平文 m は 1024bit とし，M より小さいものとする．暗号化においては新たに生成した二つの乱数 r_1, r_2 を使用し，暗号文 $C = (C_1, C_2)$ を次のように生成する．

$$C_1 = m \cdot g_1^{r_1} \ (\mathrm{mod}\ M) \tag{3}$$

$$C_2 = m \cdot g_2^{r_2} \ (\mathrm{mod}\ M) \tag{4}$$

受け取った暗号文 $C = (C_1, C_2)$ を復号するには，秘密鍵 $\{p, q\}$ を使い，次の変換を施す．

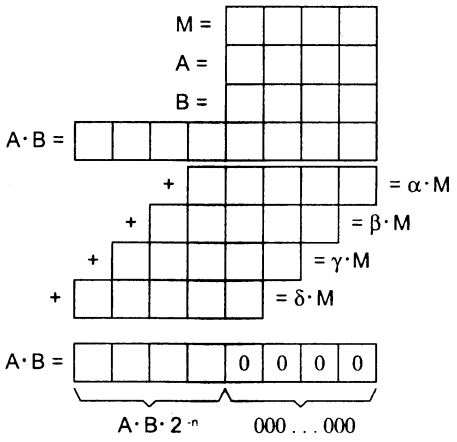


図 1 モンゴメリ乗算のアイデア

$$m = C_1 Q q + C_2 P p \ (\mathrm{mod}\ M) \tag{5}$$

ここで P, Q はそれぞれ $Qq \equiv 1 \ (\mathrm{mod}\ p)$, $Pp \equiv 1 \ (\mathrm{mod}\ q)$ を満たす．

3. 剰余乗算の実現方法

暗号および復号の基本演算は多ビット整数 M を法とした剰余乗算である．ここでは冗長表現に基づく剰余乗算とこれに基づく復号化回路の実現方法について述べる．

3.1 モンゴメリ乗算

剰余乗算を実現するには通常除算が必要となり，計算時間が非常に大きくなる．モンゴメリ乗算では剰余演算の特性を利用し，剰余乗算を加算のみで行なうことのできるアルゴリズムである．正確には $(n+1)$bit の M を法とする世界で $(A \times B \times 2^{-n})$ を効率良く計算するアルゴリズムである (演算のためには $gcd(M, 2) = 1$ を満たす必要があるが，本稿で扱う問題においては常に満たされている)．

モンゴメリ乗算の基本的なアイデアを図 1 に示す．図は 4bit×4bit の例であり，4 度の加算が順に行なわれている．繰返しのなかでは，条件によって M を加えることで中間結果の LSB が必ず 0 になるように処理が行なわれる．

モンゴメリ乗算においては通常正規化が必要となる．しかし，$B' = B \times 2^n \ (\mathrm{mod}\ M)$ なる数 B' をあらかじめ計算しておき，B のかわりに使用することで，正規化の操作を省くことができる．

法の数 M が 1024bit である場合の疑似コードを図 2 に示す．ループの中では，中間結果 S' がまず始めに計算される．もし S' の LSB が 1 であった場合，S' の LSB を 0 に置き換えるため，S' に M が加えられる．$S' + L_0 M$ の結果は右に 1bit シフトされるため，S のビット長は常に 1025bit となる．

3.2 冗長表現の利用

多くの暗号システムでは法 M として 1024bit や 2048bit といった非常に大きな整数が使われる．そのためモンゴメリ乗算における加算には大きな遅延が伴う．そこで多ビット整数

```
MONT (A,B,M)
{
    S := 0;
    for i = 0 to 1023
    {
        S'  := S + b0A;
        L0  := S' mod 2;
        S   := (S' + L0M) / 2;
        B   := B / 2;
    }
    If S ≧ M then S := S - M;
}
```

図 2　モンゴメリ乗算の疑似コード

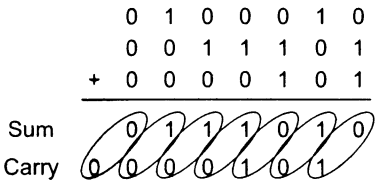


図 3　桁上保存加算

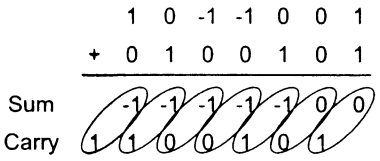


図 4　冗長二進加算

の加算を現実的な時間で行なう二つの方法について考えた．一つは Carry Save Adder(桁上保存加算)，そしてもう一つは Redundant Binary Adder (冗長二進加算) である．どちらの加算方式も冗長な数値表現を使うため，キャリーの伝搬が不必要である．

図 3 は CSA の基本的な動きを示している．各桁について 3 つのビットがそれぞれ足され，SUM と CARRY が生成される．そのため CSA は全加算器をアレイ状に並べることで実現でき，遅延の大きな原因となるキャリー線は存在しない．

図 4 は RBA の基本的な動きを示しており，各桁はそれぞれ {1,0,-1} のいずれかで表現されている．RBA の中では，まず中間 SUM と中間 CARRY を求め，これらを足し合わせることで最終的な結果を得る．ただし本稿においては，暗号の性質上，加数が必ず正であるので，通常の RBA とは少し異なる RBA を導入した．加数が必ず正であるため，-1 のキャリーが発生しない．そのため中間 SUM は必ず {0,-1} のどちらかとなり，中間 CARRY は {0,-1} のどちらかとなる．本稿で導入した冗長二進加算の真理値表を表 5 に示す．

最後に CSA/RBA を利用したモンゴメリ乗算回路を図 6 に，そのデータフローを図 7 に示す．

図 5　冗長二進加算の真理値表

A	0	0	1	1	-1	-1
B	0	1	0	1	0	1
中間 Sum	0	-1	-1	0	-1	0
中間 Carry	0	1	1	1	0	0

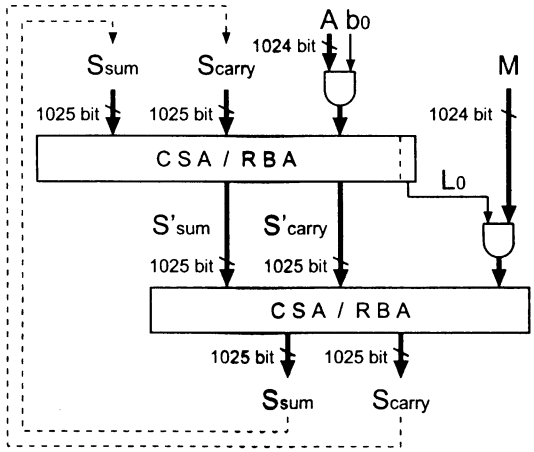


図 6　CSA/RBA ベースのモンゴメリ乗算

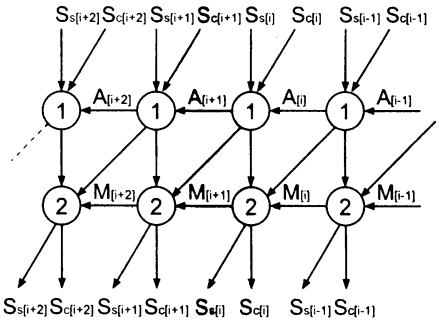


図 7　データフロー

3.3　復号化回路のアーキテクチャ

式 (5) にある通り，復号化部分は剰余乗算 2 回と加算 1 回のみで構成されており，RSA 暗号にくらべて計算量は非常に少ない．復号化処理を演算別にわけて記述すると

$$X = C_1 \times \zeta \tag{6}$$

$$Y = C_2 \times \eta \tag{7}$$

$$m = X + Y \pmod{M} \tag{8}$$

となる．ここで ζ, η は式 (5) 中の Qq, Pp に相当し，あらかじめ計算しておくことができる．そこでモンゴメリ乗算を図 8 に示すよう改良することで，復号化処理を実現できる．

さらに，ζ, η はどちらも定数なので，$\xi(=\zeta+\eta)$ をあらかじめ計算しておくことで式 (6) と式 (7) の演算で乗算器を共有することができる．すなわち (C_{1_0}, C_{2_0}) に対して $(0,0)$ なら何もせず，$(0,1)$ なら ζ を，$(1,0)$ なら η を，$(1,1)$ なら ξ を加える．そ

```
mMONT (C1,C2,ζ,η,M)
{
    S := 0;
    for i = 0 to 1023
    {
        S' := S + C1oζ + C2oη;
        L0 := S' mod 2;
        S  := (S' + L0M) / 2;
        C1 := C1 / 2;
        C2 := C2 / 2;
    }
    if S ≧ M then S := S - M;
}
```

図 8 改良モンゴメリ乗算

の後 M で modulo をとる.

for の繰返し後には, 答として $C_1 \times \zeta + C_2 \times \eta \pmod M$ を表す 2 つの数を得るため, この 2 つの数字を通常の二進数に直すための後処理が必要である. 更に, 得られたものが M より大きな場合, 減算が必要となる. この変換と減算は以下の式で実現することができる (ただし $m_{\rm sum}, m_{\rm carry}$ は乗算の結果である)

$$m' = m_{\rm sum} + m_{\rm carry} \tag{9}$$

$$m'' = m' - M \tag{10}$$

$$\textbf{if } (m'' < 0) \quad \{m = m'' + M\} \tag{11}$$

$$\textbf{else} \quad \{m = m''\} \tag{12}$$

4. 暗号 LSI のアーキテクチャ

3 節において, CSA/RBA に基づくモンゴメリ乗算を使い復号化処理を実現できることを示した. 本節では, 暗号化された動画像をリアルタイムに復号化することを目標とした暗号 LSI のアーキテクチャの概要と高速化手法について述べる.

目標とする動画像は 640×480 pixel -フルカラー (24bit) - 30 フレーム/sec, とした. いいかえると 1/13.5M 毎に 24bit のデータが送られてくることになる. 目標の動画像をリアルタイムに処理するためには, 324Mbps のスループットが必要となる. 復号化処理の中では 1024bit が 1 データブロックとして扱われているため, 1 つのデータブロック中には 42clock 分のデータが含まれている. よって, リアルタイムで復号化処理を行なうには, 1 ブロックを 3185ns (= 1/13.5M × 42clock) 以内に処理する必要がある. この時間的制約を満たすために (1) 加算ループの展開, (2) パイプライン化を行なった.

4.1 加算ループの展開

復号化処理におけるモンゴメリ乗算では CSA/RBA による加算を 1024 回繰返すことによって解を得る. 繰返し回数が非常に多いので, 回路を 100MHz で動作させても復号化処理を 3185ns 以内に終らせることが難しい.

剰余乗算モジュールでは, 繰返し回数を減らすために, 通常の乗算器を 4 つ直列につないだモジュールを設計した. つまり, for ループを (i=0; i<1024; i=i+1) ではなく (i=0; i<1024; i=i+4) と変更したことになる. そのために新たに CSA/RBA を必要とするが繰返し回数を 256 回に減らすことができた. 実際の設計では 4 つの同じ CSA/RBA モジュールを図 9 の上部に示すように配置した.

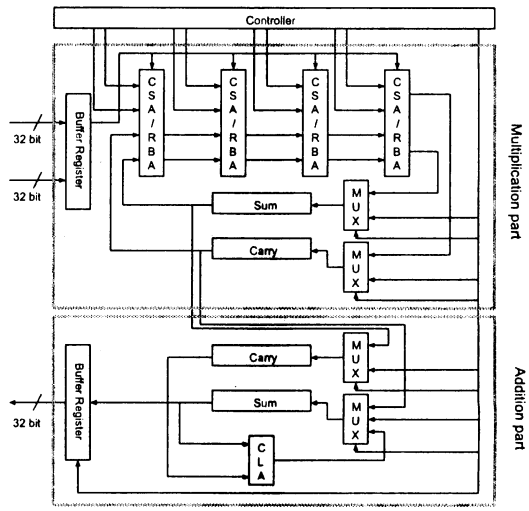


図 9 復号化回路のブロック図

4.2 パイプライン化

モンゴメリ乗算により得られる解は冗長な表現形式であるので, 通常の二進数に戻すための後処理が必要となる. このためにはキャリーつきの加算を行なわなければならない. 条件によってはさらに減算が必要となる. そこで剰余乗算モジュール (図 9:上部) と別に, 後処理を行なう加算モジュール (図 9:下部) を独立に構成し, 二つのモジュールが並列に動くようにした.

加算モジュールでは, 剰余乗算によって 1024bit の整数が 2 つ ($m_{\rm sum}, m_{\rm carry}$) が得られるので 16bit の加算器を繰返し 64 回使うことで変換を実現した. 次に, この計算結果の二進数から M を引く処理を行なう. これも 64 クロック必要である. さらに, この結果が負になった場合は, M を加えることで元の数を得る. 最悪の場合でも 192 クロックで終了するので, 剰余乗算モジュールで必要な 256 クロックと釣り合う.

4.3 暗号化について

本稿では復号化に重点をおいた LSI を設計したが, これを複数使用することでデータを暗号化することもできる. 暗号化は式 (3)(4) で定義される. ここで g_1^{r1} および g_2^{r2} はあらかじめ計算しておくことができる. そこで ζ(式 (6)) もしくは η(式 (7)) を 0 としておくことにより LSI をひとつの乗算器とみなすことができる. そこで, 本 LSI を図 10 に示すように 2 つ用いることで暗号化を実現できる. なお実際の暗号化においては, 乱数 $r1, r2$ を変更する処理が必要であり, 付加的なハードウェアを必要とする. ただし, 変更の間隔が十分長ければ, 付加的なハードウェアの量はそれほど大きくならない.

5. LSI の実装と評価

本稿で提案したアーキテクチャーに基づく回路を Verilog

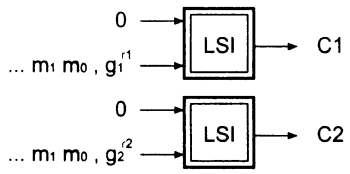


図 10　暗号化の実現方法

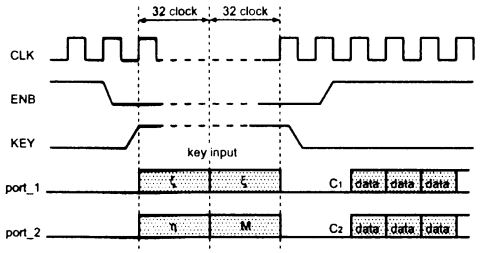
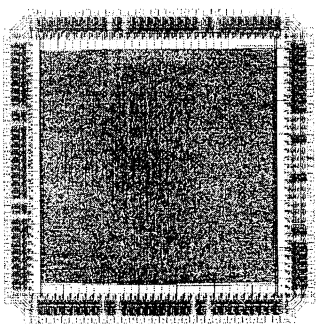


図 11　タイミングチャート

HDL を用いて設計した. 設計においては CSA/RBA の両方のタイプを設計した. いずれも 1024bit の鍵長を基本としている. なお, 設計自体はビット長をパラメータ化しているので, 鍵長 1024bit の回路を設計する前に, さまざまな鍵長のシステムを設計し, その面積評価を行なった. 表 1 に論理合成の結果を示す. 評価には合成ツールとして dc_shell, ライブラリは VDEC EXD Rohm0.35μm ライブラリを用いた. この結果から CSA/RBA いずれの構成をとった場合もほぼ同じ面積かつ速度となることがわかった.

図 11 は復号化回路における初期化部分のタイミングチャートである. 最初に復号化に必要な鍵がレジスタにセットされる. 必要とされるのは $\{\zeta, \eta, \xi, M\}$ の 4 つ (いずれも 1024bit) である. ENB と KEY がそれぞれ 0, 1 になったとき, キーの入力が開始される. 入力ポートが 32bit のため, キー入力には 64clock (1024bit×4 = 32bit×2-port×64clock) が必要となる. そして ENB と KEY がそれぞれ 1, 0 となったときに復号化処理が開始される. 本システムでは入力される暗号文の長さは可変であり, MODE ビットにより切替えることが可能である. 暗号文入力のスループットはこれらのビット長に反比例する.

鍵長 256bit-CSA のタイプについては Rohm0.35μm プロセス-4.9mm×4.9mm むけにレイアウトを行ない, 実際にチップを作成した (図 12). そして出来上がったチップを用いて, 暗号化された動画像信号を遅延なしに復元できることを実環境下で確認した. 確認においては MMS 社製の MU200-SXCW FPGA ボードを用いた. 本ボードは VDEC Rohm0.35μm 160pin パッケージの LSI の動作テストを行なうためのもので, LSI と FPGA(Altera) の協調動作が可能である. また, 別基盤を用いて動画像のリアルタイム処理の実験を行なうことができる. 実験では, FPGA 側で暗号化を行ない, LSI 側で復号して, リアルタイムに動画像の暗号・復号の処理ができることを示した.



IO Pin	103
鍵長	256 bit
動作周波数	81 MHz
最長パス	10.17 ns
コアの面積	3.6×3.6 mm^2
入力データ長	8,16,24,32 bit

図 12　試作した LSI とスペック

6. ま と め

本稿では RSA 暗号を基に開発された新暗号方式に対し, これを効率的に実現できる LSI アーキテクチャを示した. 演算部分は改良モンゴメリ乗算, CSA/RBA タイプの 4-2 加算器, CLA からなっている. CSA/RBA はビット長によらず遅延が一定なので, 多ビットの加算に適していることを示した. そして鍵長 256bit のシステムを Rohm0.35μm プロセス上で LSI 化した. 設計した LSI は 81MHz で動作し, 暗号化された 640×480 pixel - フルカラー (24bit) - 30 フレーム/sec の動画像 (=324Mbps) をリアルタイムに復号化できることを実験によって確認した.

今後は, アーキテクチャをさらに改良し, よりスループットが高く長い鍵長をもつ復号化 LSI を設計することを計画している. またリアルタイムの暗号化についても本 LSI の応用を考えている.

謝　辞

本研究を進めるにあたり日頃から有益なご助言, ご指導を頂いた早稲田大学大学情報生産システム研究科の吉村猛教授, 渡邊孝博教授に深く感謝します. また, チップ試作については東京大学大規模集積システム設計教育研究センターを通し, シノプシス株式会社の協力で行なわれたものである.

文　献

[1] Douglas R. Stinson, "Cryptography Theory and Practice," CRC Press, Inc, February 2002.

[2] Hatsukazu Tanaka, "A New Simple Public-Key Cryptosystem and Its Application to Digital Signature," in *Proc. SCIS 2003*, January 2003.

[3] Alan Daly and William Marnane, "Efficient Architectures for Implementing Montgomery Modular Multiplication and RSA Modular Exponentiation on Reconfigurable Logic," in *Proc. 10th International Symposium on FPGA*, pp.40–49, February 2002.

[4] Shimbo Atsushi, Nozaki Hanae and Kawamura Shin-ichi, "Fast RSA Computation Algorithm and LSI," Toshiba Re-

表 1 論理合成の結果

	Carry Save Adder				Redundant Binary Adder			
鍵長 (bit)	Comb.(mm^2)	Non-comb.(mm^2)	総面積 (mm^2)	遅延 (ns)	Comb.(mm^2)	Non-comb.(mm^2)	総面積 (mm^2)	遅延 (ns)
128	1.169153	0.463732	1.632885	9.03	1.294643	0.467224	1.761867	9.66
256	2.293255	0.930849	3.224104	9.66	2.635200	0.936719	3.571919	9.66
512	5.214251	1.838875	7.053126	9.54	5.254580	1.856380	7.110959	9.61
1024	10.876343	3.667536	14.539919	9.69	10.472685	3.686708	14.159393	9.65

view Vol. 56 No. 7, 2001.

[5] Alexandre F. Tenca and Cetin K. Koc, "A Scalable Architecture for Modular Multiplication Based on Montgomery's Algorithm," *IEEE Trans. on Computers*, Vol. 52, No. 9, September 2003.

[6] Ciaran McIvor, Maire McLoone and John V MvCanny, "Fast Montgomery Modular Multiplication and RSA Cryptographic Processor Architectures" *37th Asilomar Conference on Signal, Systems & Computers*, Vol. 1, pp.379–384, November 2003.

[7] Koon-Shik Cho, Je-Hyuk Ryu and Jun-Dong Cho, "High-Speed Modular Multiplication Algorithm for RSA Cryptosystem," in *Proc.* IECON'01, Vol. 1, pp.479–483, Dec 2001.

[8] Thomas Blum and Christof Paar, "High Radix Montgomery Modular Exponentiation on Reconfigurable Hardware," *IEEE Trans. on Computers*, Vol. 50, No. 7, pp.759–764, July 2001.

[9] Akashi Satoh and Kohji Takano, "A Scalable Dual-Field Elliptic Curve Cryptographic Processor," *IEEE Trans. on Computers*, Vol. 52, No. 4, April 2003.

[10] Naofumi Takagi and Shuzo Yajima "Mudular Multiplication Hardware Algorithms with a Redundant Representation and Their Application to RSA Cryptosystem," *IEEE Trans. on Computers*, Vol. 41, No. 7, pp.887–891, July 1992.