情報処理学会論文誌 Vol. 51 No. 9 1847–1858 (Sep. 2010)

推薦論文

# RSA 暗号プロセッサ自動生成システムの設計と評価

馬場祐一[†1] 宮本篤志[†1] 本間尚文[†1]
青木孝文[†1] 佐藤証[†2]

本稿では，高基数モンゴメリ乗算に基づく RSA 暗号プロセッサアーキテクチャを提案し，その RSA 暗号プロセッサを設計仕様に応じて自動生成するジェネレータの性能評価を示す．提案アーキテクチャは，高いスケーラビリティを有しており，多様な基数や算術アルゴリズムと組み合わせることが可能である．本ジェネレータは，提案する 2 種類のアーキテクチャに対して，5 種類の基数および 84 種類の積和演算アルゴリズムを組み合わせることで計 840 種類の RSA 暗号プロセッサを生成できる．90 nm CMOS ライブラリを用いた RSA 暗号プロセッサの合成結果から，提案アーキテクチャおよびジェネレータの有効性を評価する．

# Design and Evaluation of RSA Processor Generation System

Yuichi Baba,[†1] Atsushi Miyamoto,[†1]
Naofumi Homma,[†1] Takafumi Aoki[†1]
and Akashi Satoh[†2]

This paper proposes two RSA processor architectures based on the high-radix Montgomery Multiplication and evaluates an RSA processor generator which provides optimized RSA processors satisfying various user requirements. The proposed architectures achieve high scalability and can be combined with various radices and arithmetic components. A total of 840 designs are obtained by the generator combining 2 architectures with 5 radices and 84 arithmetic components. We synthesized all the processors with a 90 nm CMOS standard cell library and evaluated the performances of the proposed architectures and the processor generator.

## 1. まえがき

身の回りのあらゆる情報機器がネットワークを介して結合されるユビキタス情報社会においては，個人情報の保護や高信頼な電子商取引が必須であり，情報セキュリティをいかに構築するかが重要な課題となる．暗号はその中核を担う技術の 1 つであり，特に，鍵配送および鍵管理，認証等に適した公開鍵暗号方式は，その基盤技術として期待されている．RSA 暗号に代表される公開鍵暗号の演算は，一般に安全性確保のために膨大な多倍長演算を必要とし，その計算コストは非常に大きなものとなる．そのため，特にリソースの限られる IC カードや組み込み用途においては，公開鍵暗号の専用ハードウェアによる実装が求められている．

RSA 暗号の暗号化・復号は，多倍長のべき乗剰余演算を基本とする．そのため，計算資源の限られたハードウェア実装では，演算を小さな処理単位に分割する多倍長演算アルゴリズムが有効となる．RSA 暗号プロセッサでは，この多倍長演算に乗剰余演算の効率的な計算手法として知られるモンゴメリ乗算[1] を組み合わせた回路アーキテクチャが数多く提案されている[2)–8)]．それらはオペランドの分割幅を表す “基数” により基数 2 と高基数のアーキテクチャに大別される．基数 2 のアーキテクチャ[3),6)] は，片方のオペランドをビットごとに分割して処理するため，加算器を用いた構成となる．シンプルな回路構成で，1 サイクルの遅延時間を小さくできるといった利点がある．これに対して，演算の基数を $2^8$，$2^{16}$，$2^{32}$ と大きくした高基数のアーキテクチャは，基数 2 のアーキテクチャと比べて高効率な回路を実現できる．文献 2) では，64 ビット乗算に基づくアーキテクチャが提案されている．

しかし，従来設計のほとんどは特定の性能をある条件下で最適化したものであり，実装するデバイスや使用するライブラリ，連携するシステムといった仕様や制約条件に柔軟に適用することは困難であった．従来の基数 2 アーキテクチャには構造が一意に決まっていて適用自体が不可能な場合も多い．また，高基数のアーキテクチャ設計[2)] も，その多倍長演算アルゴリズムに主眼が置かれ，ある特定の設計とその実装結果を示すにとどまっていた．高基数アルゴリズムの特長を十分に発揮するためには，その性能を左右するデータパスを設計仕様

†1 東北大学大学院情報科学研究科
Graduate School of Information Sciences, Tohoku University
†2 産業技術総合研究所
National Institute of Advanced Industrial Science and Technology
本論文の内容は 2009 年 9 月の FIT2009 第 8 回情報科学技術フォーラムにて報告され，同プログラム委員長により情報処理学会論文誌ジャーナルへの掲載が推薦された論文である．


© 2010 Information Processing Society of Japan

や制約条件に応じてアーキテクチャから算術アルゴリズムまで適切に設計することが必要になる．近年，スマートカードに代表される小型機器から高速処理が求められるサーバ等の機器に至るまで公開鍵暗号の用途は多様化しており，要求される性能に応じた系統的な設計システムの重要性はますます高まると予想される．

以上の背景から，本稿では，公開鍵暗号モジュールの系統的設計システムとして RSA 暗号プロセッサジェネレータを提案する．本ジェネレータは，演算の基数，アーキテクチャおよび算術アルゴリズムという 3 つの設計パラメータの設定により，高基数モンゴメリ乗算に基づく 840 種類の RSA 暗号プロセッサを自動生成する．これらの設計パラメータを系統的に選択することで，仕様や制約条件に応じた適切なプロセッサを設計することが可能となる．特に，データパスの演算器構成が異なる 2 種類の高基数モンゴメリ乗算アーキテクチャを新たに導入することにより，演算時間，回路面積，消費電力および演算効率の面で，従来の最適設計と同等以上の性能を得ることができる．また，本稿では，840 種類すべての RSA 暗号プロセッサを生成した性能評価実験を通して，本ジェネレータの有効性を示す．より実装に近い評価を行うため，90 nm CMOS スタンダードセルライブラリを用いた合成後の配置配線結果から，演算時間，回路面積および消費電力を見積もった．本実験により，1,024 ビットの RSA 暗号に対して 1.25 [Kgate]@174.89 [ms] の小型実装（基数 $2^8$）から 227.16 [Kgate]@1.45 [ms] の高速実装（基数 $2^{128}$）まで多様な回路性能が連続的に得られることを示す．

## 2. RSA 暗号

### 2.1 べき乗剰余演算

RSA 暗号は，べき乗剰余演算により暗号化・復号処理を行う公開鍵暗号である．$P$ を元のデータ（平文），$C$ を暗号文，$E$ と $N$ を公開鍵，$D$ を秘密鍵とすると，暗号化および復号処理は次のような式で表される．

$$C = P^E \bmod N \tag{1}$$

$$P = C^D \bmod N \tag{2}$$

公開鍵である法 $N$ や秘密鍵 $D$ には，安全性の観点から 1,024 ビット以上の多倍長整数が利用され，また，平文 $P$ や暗号文 $C$ も同一の語長が用いられる．

RSA 暗号のべき乗剰余演算は，指数 $E$ あるいは $D$ のビットパターンに応じて，乗剰余演算（自乗剰余算と乗剰余算）を繰り返すことによって実現される．ALGORITHM 1 に，最も基本的なべき乗剰余演算アルゴリズムである左バイナリ法を示す．左バイナリ法では，

**ALGORITHM 1**
Modular Exponentiation (MSB First).

| | | |
|---|---|---|
| Input: | $X$, $N$, $E = (e_{k-1}, ...e_1, e_0)_2$ | |
| Output: | $Z = X^E \bmod N$ | |
| 1： | $Z := 1$; | |
| 2： | **for** $i = k-1$ **downto** 0 | |
| 3： | $Z := Z \cdot Z \bmod N$; | – squaring |
| 4： | **if** $(e_i = 1)$ **then** | |
| 5： | $Z := Z \cdot X \bmod N$; | – multiplication |
| 6： | **end if** | |
| 7： | **end for** | |

**ALGORITHM 2**
Montgomery Multiplication.

| | | |
|---|---|---|
| Input: | $X, Y, N$, $W = -N^{-1} \bmod R$ | |
| Output: | $Z = XYR^{-1} \bmod N$ | |
| 1： | $t := XY \cdot W \bmod R$; | |
| 2： | $Z := (XY + tN)/R$; | |
| 3： | **if** $(Z > N)$ **then** $Z := Z - N$; | |

指数ビットの左側（最上位ビット）から始めて，ビットが 0 ならば自乗剰余算（3 行目），1 ならば自乗剰余算と乗剰余算（3–6 行目）をペアで行う．

RSA 暗号実装では，べき乗剰余演算中に繰り返される乗剰余演算をいかに効率的に実現するかが重要となる．乗剰余演算アルゴリズムの中でも，ハードウェアおよびソフトウェア実装の双方に有効なものとしてモンゴメリ乗算アルゴリズム[1] がよく知られている．

### 2.2 モンゴメリ乗算

モンゴメリ乗算では，2 つの整数 $X$，$Y$ に対して，次の演算を行う．

$$Z = XYR^{-1} \bmod N \tag{3}$$

ここで，$X$，$Y$，$R$，$N$ は以下の関係を持つ．

$$0 \le X, Y < N < 2^k = R \tag{4}$$

ALGORITHM 2 に，基本的なモンゴメリ乗算アルゴリズムを示す．本アルゴリズムで


© 2010 Information Processing Society of Japan

**ALGORITHM 3**
HIGH-RADIX MONTGOMERY MULTIPLICATION.

| | | |
|---|---|---|
| Input: | $X = (x_{m-1}, ..., x_1, x_0)_{2^r}$, | |
| | $Y = (y_{m-1}, ..., y_1, y_0)_{2^r}$, | |
| | $N = (n_{m-1}, ..., n_1, n_0)_{2^r}$, | |
| | $w = -N^{-1} \bmod 2^r$ | |
| Output: | $Z = XY2^{-r \cdot m} \bmod N$ | |
| 1 : | $Z := 0$; | |
| 2 : | **for** $i = 0$ **to** $m - 1$ | – Loop 1 |
| 3 : | $c := 0$; | |
| 4 : | $t_i := (z_0 + x_i y_0) w \bmod 2^r$; | |
| 5 : | **for** $j = 0$ **to** $m - 1$ | – Loop 2 |
| 6 : | $q := z_j + x_i y_j + t_i n_j + c$; | |
| 7 : | **if** $(j \neq 0)$ **then** $z_{j-1} := q \bmod 2^r$; | |
| 8 : | $c := q/2^r$; | |
| 9 : | **end for** | |
| 10: | $z_{m-1} := c$; | |
| 11: | **end for** | |
| 12: | **if** $(Z > N)$ **then** $Z := Z - N$; | |

は，$XY$ の乗算結果に法 $N$ の倍数を加算し，$R$ で割り切れる値に補正することで，剰余算を不要にしている．$R$ は $2^k$ であるため除算はシフト演算により実現される．このように，モンゴメリ乗算アルゴリズムでは，演算コストの高い除算を行わずに積和演算とシフト演算で効率的に乗剰余演算を実現することができる．

RSA 暗号では，通常 1,024 ビット以上の多倍長データを扱うため，しばしばデータをワードごとに分割したアルゴリズムが用いられる．本稿では，その中でも演算効率に優れた高基数のモンゴメリ乗算アルゴリズム[2] を考える．

高基数モンゴメリ乗算アルゴリズムでは，基数によってオペランドのビット幅 $k$ は，ビット長 $r$ のワードごとに $m$ 分割（$m = k \cdot r$）される．たとえば，入力 $X$ はワード $x_i$（$0 \leq i \leq m-1$）によって以下のように表される．

$$X = x_{m-1} \cdot 2^{r(m-1)} + ... + x_1 \cdot 2^r + x_0 \tag{5}$$

本稿では，式 (5) を以下のように簡略化する．

$$X = (x_{m-1}, ..., x_1, x_0)_{2^r} \tag{6}$$

高基数モンゴメリ乗算アルゴリズムを ALGORTIHM 3 に示す．$k$ ビットの入力 $X$，$Y$，

**ALGORITHM 4**
MODULAR EXPONENTIATION WITH *MontMult*.

| | | |
|---|---|---|
| Input: | $X$, $N$, | |
| | $E = (e_{k-1}, ..., e_1, e_0)_2$ | |
| Output: | $Z = X^E \bmod N$ | |
| 1 : | $W := -N^{-1} \bmod R$; | |
| 2 : | $Y := XR \bmod N$; | |
| 3 : | $Z := R \bmod N$; | |
| 4 : | **for** $i = k - 1$ **downto** 0 | |
| 5 : | $Z :=$ *MontMult* $(Z, Z, N, W)$; | – squaring |
| 6 : | **if** $(e_i = 1)$ **then** | |
| 7 : | $Z :=$ *MontMult* $(Z, Y, N, W)$; | – multiplication |
| 8 : | **end if** | |
| 9 : | **end for** | |
| 10: | $Z :=$ *MontMult* $(Z, 1, N, W)$; | |

$N$ は，それぞれ $r$ ビットごとのワード $x_i$，$y_j$，$n_j$（$0 \leq i$，$j \leq m-1$）に分割され，Loop 1（$x_i$ に対するループ）と Loop 2（$y_j$，$n_j$ に対するループ）により，繰り返し演算される．ここで，一時変数 $q$ は $2r$ ビットであり，上位 $r$ ビット，下位 $r$ ビットがそれぞれ中間キャリー $c$，中間和 $z$（$0 \leq j \leq m-1$）となる．このとき，7 行目において，中間和を 1 つ前のワード（すなわち $z_{j-1}$）に格納することで，ALGORITHM 2 におけるシフト演算も同時に実行する．最終的に，演算の終了時に $Z = (z_{m-1}, ..., z_1, z_0)_{2^r}$ に格納された値が出力となる．

ALGORITHM 4 はモンゴメリ乗算に左バイナリ法を組み合わせたアルゴリズムである．関数 *MontMult* は，式 (3) で表されるモンゴメリ乗算であり，1，2 行目はモンゴメリ乗算を適用する際に必要となる前処理の演算を表す．

## 3. RSA 暗号プロセッサ

### 3.1 モンゴメリ乗算回路アーキテクチャ

本稿では，多様な性能の RSA 暗号プロセッサを実現するため，高基数モンゴメリ乗算アルゴリズムを基本とする 2 種類の回路アーキテクチャ（Type-I と Type-II）を提案する．高基数モンゴメリ乗算のハードウェア実装では，ALGORITHM 3 の 6 行目にある積和演算がクリティカルパスとなり，その演算方法が回路性能に大きな影響を与える．そこで，中間データの伝搬方式に着目し，アーキテクチャ内部の積和演算器構成が異なる Type-I と


© 2010 Information Processing Society of Japan

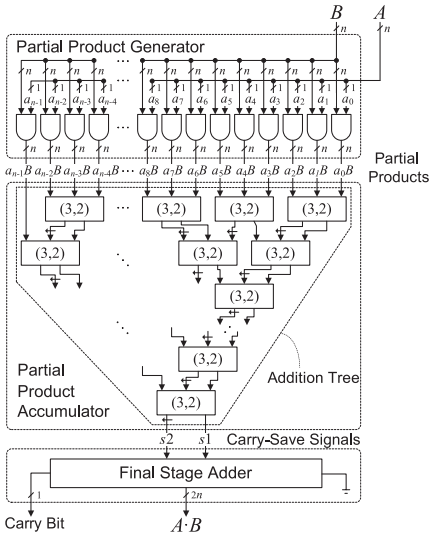



**図 1**　$n$ ビット並列乗算器の構成
Fig. 1　$n$-bit parallel multiplier.

Type-II を考える．

まず，提案アーキテクチャのアイデアを説明するため，**図 1** に基本的な $n$ ビット並列乗算器の構造を示す．乗算器や積和演算器は，部分積を生成する PPG（Partial Product Generator），生成された部分積を累積加算する PPA（Partial Product Accumulator）および，Carry-Save 形式で表された PPA の 2 出力を桁上げ伝搬加算する FSA（Final Stage Adder）から主に構成される．**図 2** に 32 ビット並列乗算器の遅延時間を桁ごとにプロットしたものを示す．横軸は出力の桁番号（$32 \times 32$ ビット乗算のため 64 ビット），縦軸は遅延時間である．図中の△および□は，それぞれ PPA および FSA における各桁の遅延時間を表す．図より，PPA では部分積の重なる 32 ビット付近に最も大きな遅延が生じていることが確認できる．一方で，FSA では桁番号が大きくなるにつれて遅延時間が大きくなっており，その桁上げ伝搬による遅延時間が大きな割合を占めている．提案アーキテクチャでは，PPA の出力である Carry-Save 形式を利用することで，この FSA による遅延時間を削減し，全体のクリティカルパスを低減させる．

**図 3** に提案する 2 種類の回路アーキテクチャ（Type-I と Type-II）を示す．$k$ ビットのオペランドをワードごとに分割して処理するため，データのバス幅は $r$ ビットとなる．各データパスは，積和演算器である Arithmetic Core およびその入力を保持するレジスタ，マルチプレクサ等から構成される．ALGORITHM 5 と 6 にそれぞれ Type-I と Type-II の回路アーキテクチャに対応する高基数モンゴメリ乗算アルゴリズムを示す．それらは Carry-Save 形式の違いにより FSA のオペランドサイズが異なる．モンゴメリ乗算の基数を $2^r$ とした場合，Type-I と Type-II における FSA のオペランドサイズは，それぞれ $2r$，$r$ ビットとなる．ここで，アルゴリズム中の $(c, z)$ は $c \cdot 2^r + z$ を表し，$w$ および $v$ はそれぞれ前処理演算 $-N^{-1} \bmod 2^r$ と $i$ ループ間のキャリーを表す．また，Carry-Save 形式は Carry-Save 信号（$cs1$，$cs2$，$ec$）と加算記号 “+” を用いて $cs1 + cs2 + ec$ と記述する．

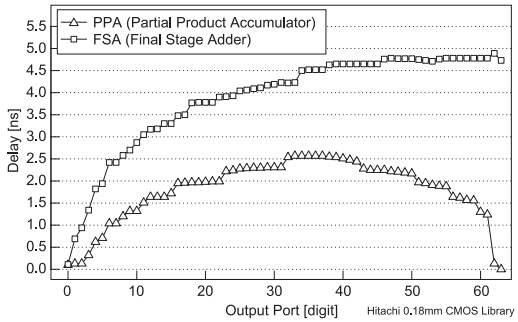
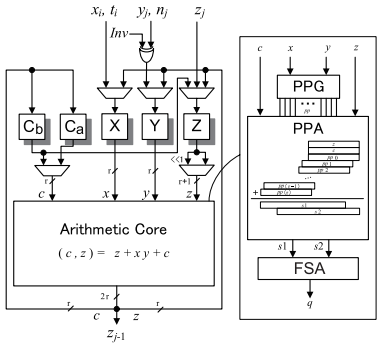



**図 2**　32 ビット並列乗算器の遅延プロファイル
Fig. 2　Delay profile on 32-bit parallel multiplier.

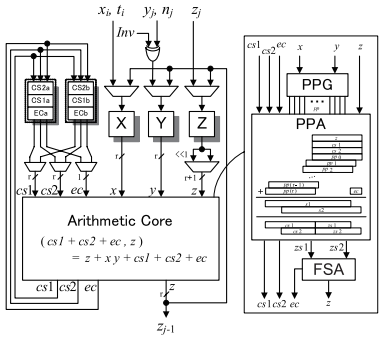
Type-I は，Arithmetic Core に基本的な 3 項積和演算器を用いたアーキテクチャである．このとき，積和演算は乗算ステップ（$(c_a, z) := z_j + x_i y_j + c_a$）および還元ステップ（$(c_b, z) := z_j + t_i n_j + c_b$）に分割され，FIOS（Finely Integrated Operand Scanning method）[9] の演算方式に従って交互に実行される．データパス内部には，Arithmetic Core の入力を保持するため，5 つのレジスタ（Ca，Cb，X，Y，Z）がある．Ca と Cb レジスタは中間キャリー $c_a$ と $c_b$，X レジスタはオペランド $x_i$，$t_i$ および $v$，Y レジスタはオペランド $y_j$，$n_j$ および $w$，Z レジスタは中間和 $z_j$ を格納する．Type-I アーキテクチャは，最低限のレジスタ数で構成可能となるが，最大 $2r$ ビットの入力を持つ FSA により遅延時間は大きくなる．ALGORITHM 5 を実行するのに必要となるサイクル数は下記の式で表される．

$$2m^2 + 4m + 1 \tag{7}$$


© 2010 Information Processing Society of Japan




(a) Type-I




(b) Type-II

**図 3**　高基数モンゴメリ乗算回路アーキテクチャ
Fig. 3　High-radix Montgomery Multiplier architectures.

ここで，$m$ は分割数である．後述する Type-II と比べて，サイクル数を少なくできるため，高速な積和演算器を適用した場合に RSA 暗号全体の処理時間を小さくすることができる．また，レジスタが少ないため，回路面積に優れたアーキテクチャともいえる．

一方，Type-II アーキテクチャは，よりバランスを重視した構成となる．積和演算の出力のうちキャリー側出力（$cs1$，$cs2$）を Carry-Save 形式のまま Arithmetic Core にフィードバックし，中間和側の出力（$zs1$，$zs2$）のみを FSA により加算する．FSA では，$r$ ビットの中間和 $z$ に加えて，1 ビットのキャリー信号 $ec$ も生成する．その結果，Type-II では

**ALGORITHM 5（Type-I）**

```
1 :  Z := 0;    v := 0;
2 :  for i = 0 to m − 1
3 :    (c_a, z_0) := z_0 + x_i y_0;
4 :    t_i := z_0 w mod 2^r;
5 :    (c_b, z_0) := z_0 + t_i n_j;
6 :    for j = 1 to m − 1
7 :      (c_a, z_j) := z_j + x_i y_j + c_a;
8 :      (c_b, z_{j−1}) := z_j + t_i n_j + c_b;
9 :    end for
10:    (v, z_{m−1}) := c_a + c_b + v;
11:  end for
12:  if (Z > N) then Z := Z − N;
```

**ALGORITHM 6（Type-II）**

```
1 :  Z := 0;    v := 0;
2 :  for i = 0 to m − 1
3 :    (cs1_a + cs2_a + ec_a, z_0) := z_0 + x_i y_0;
4 :    t_i := z_0 w mod 2^r;
5 :    (cs1_b + cs2_b + ec_b, z_0) := z_0 + t_i n_j;
6 :    for j = 1 to m − 1
7 :      (cs1_a + cs2_a + ec_a, z_j)
                  := z_j + x_i y_j + cs1_a + cs2_a + ec_a;
8 :      (cs1_b + cs2_b + ec_b, z_{j−1})
                  := z_j + t_i n_j + cs1_b + cs2_b + ec_b;
9 :    end for
10:    (v, z_{m−1})
         := cs1_a + cs2_a + ec_a + cs1_b + cs2_b + ec_b + v;
11:  end for
12:  if (Z > N) then Z := Z − N;
```

Type-I の基本的なレジスタに加えて中間キャリー $cs1$，$cs2$，$ec$ 用に別途レジスタ（CS1，CS2，EC）が必要となる．キャリー用のレジスタ数は Type-I に比べて大きくなるが，FSA のオペランド長が小さくなるため，FSA のサイズを抑えることができる．また，それにともない，Arithmetic Core の遅延時間も Type-I に比べて削減される．ALGORITHM 6 を実行するのに必要なサイクル数は下記の式で表される．


© 2010 Information Processing Society of Japan

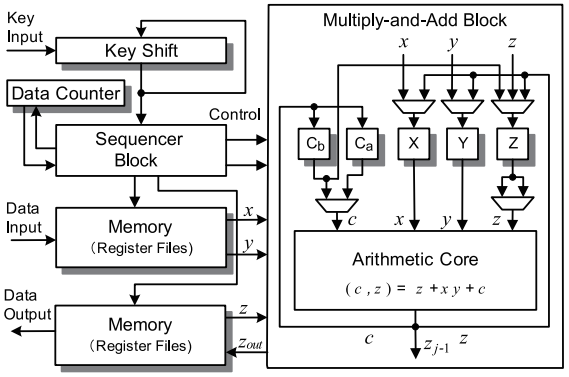



**図 4**　RSA 暗号プロセッサ
Fig. 4　RSA cryptographic processor.

$$2m^2 + 5m + 1 \tag{8}$$

Type-I に比べ，$(v, z_{m-1})$ の計算に 2 サイクルを要するため，全体のサイクル数はやや増加する．結果として，レジスタ数やサイクル数の増加に対して同等以上の FSA サイズと遅延時間の削減が期待できるため，演算効率（回路面積に対する演算性能）に優れたアーキテクチャといえる．

### 3.2　RSA 暗号プロセッサアーキテクチャ

**図 4** に提案する RSA 暗号プロセッサのブロック図を示す．本プロセッサは，演算部 Multiply-and-Add Block，制御部 Sequencer Block，メモリ Memory，鍵レジスタ Key Shift およびカウンタ Data Counter から構成される．Multiply-and-Add Block は，モンゴメリ乗算器のデータパスであり，上述の 2 種類により実現される．Key Shift は，$k$ ビットの鍵 $E$ を格納するシフトレジスタである．べき乗剰余演算のシーケンスに従い，鍵情報を 1 ビットずつ制御ブロックへと入力する．Data Counter は，データおよび繰返し回数のカウンタであり，値を保持する 3 つのレジスタと加算器から構成される．カウンタの値は，演算時の条件分岐判定やメモリアドレス生成に使用される．メモリは，2 つの $r$ ビットレジスタアレイから構成される．初期状態では，入力 $X$ および $N$ が $r$ ビットごとに格納される．また，Sequencer Block は，ALGORITHM 4 用の制御回路であり，べき乗剰余演算を実行するために各ステップでモンゴメリ乗算器を複数回呼び出す．

本プロセッサのクロックサイクル数 $t_c$ は，以下の式で表される．

$$t_c = t_{pre} + t_{mont} \cdot \left(\frac{3}{2}k + 1\right) \tag{9}$$

ここで，$t_{pre}$ は前処理（ALGORITHM 4 の 1，2 行目），$t_{mont}$ はモンゴメリ乗算の処理サイクル数を表す．RSA 暗号の処理サイクル数 $t_c$ は，鍵のビット値によって変化するが，ここでは 0 と 1 が半分ずつ発生するとして見積もる．そのとき，モンゴメリ乗算の処理サイクル数 $t_{mont}$ の係数は $(3k/2) + 1$ となる．また，$t_{pre}$ は，以下のように表される．

$$t_{pre} = (2r + 1) + (km + k + m + 1) \tag{10}$$

ここで，右辺の括弧で括られた部分は，それぞれ ALGORITHM 4 の 1 と 2 行目に対応する．前処理の演算は，加減算処理を基本として実現される．両アーキテクチャType-I と Type-II は同等の加減算器（図 3 中の FSA）を搭載するため，$t_{pre}$ は等しい．一方，モンゴメリ乗算の処理サイクル数は，式 (7) および式 (8) に示すように Type-I と Type-II で異なる．たとえば，鍵長 $k = 1{,}024$，分割数 $m = 32$ の場合，Type-I では，モンゴメリ乗算のサイクル数が 2,177，RSA 暗号処理のサイクル数が約 3,380 K となる．一方，Type-II では，モンゴメリ乗算のサイクル数が 2,209，RSA 暗号処理のサイクル数が約 3,429 K となる．Type-II のサイクル数は，ALGORITHM 6 の 10 行目により，Type-I に比べて 2%程増加する．

## 4.　RSA 暗号プロセッサジェネレータ

### 4.1　RSA 暗号プロセッサジェネレータの構成

本節では，提案アーキテクチャのスケーラビリティを応用した RSA 暗号プロセッサジェネレータの構成を示す．本ジェネレータは，上記 2 種類のアーキテクチャに加えて，データパスの性能を左右する演算の基数と積和演算器の算術アルゴリズムを設計パラメータとする．各設計パラメータを個別に指定することが可能であるため，従来では困難だった様々な設計仕様や制約条件に応じた適切な RSA 暗号プロセッサの生成が可能となる．

**図 5** に提案する RSA 暗号プロセッサジェネレータの構成を示す．本ジェネレータは，主に仕様入力部，演算器（データパス）生成部，コントローラ生成部およびコード合成部から構成される．仕様入力部は，各種設計パラメータ（基数，アーキテクチャ，演算器のアルゴリズム）の値に従い，演算器生成部およびコントローラ生成部に必要な情報を入力する．演算器生成部は，演算器モジュールジェネレータ（AMG: Arithmetic Module Generator）[10),11)] により実現される．AMG は，演算器の算術アルゴリズムを系統的にライブラリ化しており，


© 2010 Information Processing Society of Japan

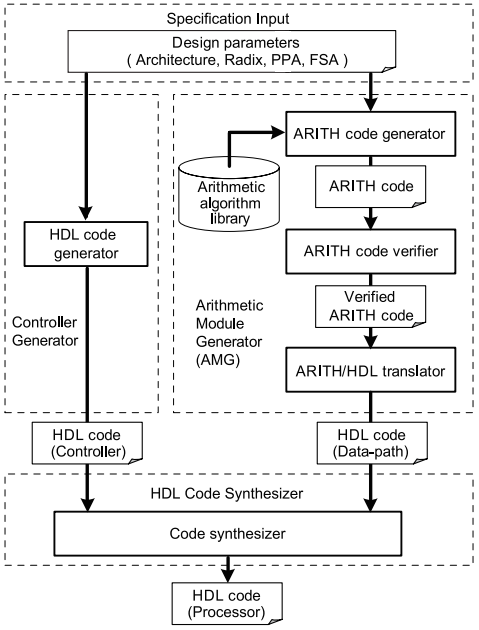



**図 5**　RSA 暗号プロセッサジェネレータ
Fig. 5　RSA cryptographic processor generator.

**表 1**　設計パラメータ
Table 1　Design parameters.

| | |
|---|---|
| Architecture | Type-I, Type-II |
| Radix | $2^8$, $2^{16}$, $2^{32}$, $2^{64}$, $2^{128}$ |
| Arithmetic Components | **Partial Product Accumulator** |
| | Array |
| | Wallace Tree |
| | Balanced Delay Tree |
| | Overturned-Stairs Tree |
| | Dadda Tree |
| | (4,2) Compressor Tree |
| | (7,3) Counter Tree |
| | **Final Stage Adder** |
| | Ripple Carry Adder |
| | Carry Lookahead Adder |
| | Ripple-Block Carry Lookahead Adder |
| | Block Carry Lookahead Adder |
| | Kogge-Stone Adder |
| | Brent-Kung Adder |
| | Han-Carlson Adder |
| | Ladner-Fischer Adder |
| | Conditional Sum Adder |
| | Carry Select Adder |
| | Fixed-Block Carry-Skip Adder |
| | Variable-Block Carry-Skip Adder |

算術アルゴリズム記述言語 ARITH [12] およびその処理系を用いて，完全に機能保証されたデータパスのコードを生成する．ARITH の詳細は文献 12) を参照されたい．一方，コントローラ生成部は，基数やアーキテクチャの情報をもとに，演算順序を制御するシーケンサや中間データを保持するメモリ等のコードを生成する．最後に，コード合成部では，生成されたデータパスおよびシーケンサのコードからプロセッサのコードを合成する．

### 4.2　生成可能な RSA 暗号プロセッサ

提案する RSA 暗号プロセッサジェネレータでは，多様な生成条件に対応するため，実用性とパレート効率性の両面から各パラメータの選択範囲を決定した．**表 1** に本ジェネレータの設計パラメータを示す．

基数（Radix）は，演算のワード長（アーキテクチャのバス幅）に対応しており，$2^8$〜$2^{128}$ の間の代表的な 5 種類（8，16，32，64 および 128 ビット幅）から選択される．基数の選択によって，回路規模とサイクル数（実行時間）といった回路の基本性能は大まかに決定される．一般に，基数が小さい場合には回路規模は小さいがサイクル数は大きくなり，基数が大きい場合にはその逆となる．基数は連携するサブシステムの仕様（バス幅）によって決まる可能性も高い．たとえば，スマートカード等の小型組み込み機器では，8，16，32 ビットの設計，高速動作を必要とするサーバ等では，32，64，128 ビットの設計を利用することが想定される．

アーキテクチャ（Architecture）は，提案する 2 種類（Type-I および Type-II）の回路アーキテクチャから選択される．上記でも述べたように，Type-I を選択した場合には，回路面積と処理時間に秀でた実装，Type-II を選択した場合には，性能のバランスのとれた演算効率に優れた実装となる．

積和演算器の算術アルゴリズム（Arithmetic Component）は，教科書的に優劣のつけられない 7 種類の PPA アルゴリズムおよび 12 種類の FSA アルゴリズムから選択される[13]．PPA のアルゴリズムは，構成要素となる桁上げ保存加算器の種類と最適化の抽象度によって


© 2010 Information Processing Society of Japan

分類される．本ジェネレータでは，一般的な (3,2) カウンタからなる Array，Wallace Tree，Balanced Delay Tree および Overturned-Stairs Tree に加え，異なる最適化レベルを持つ (4,2) Compressor Tree，(7,3) Counter Tree，Dadda Tree を用いた．一方，FSA のアルゴリズムでは，一般的に用いられる桁上げ伝搬加算器（Ripple Carry Adder，Carry Lookahead Adder，Ripple-Block Carry Lookahead Adder，Block Carry Lookahead Adder）に加え，桁上げ先見の考えを一般化した 4 種類の Parallel Prefix Adder（Kogge-Stone Adder，Brent-Kung Adder，Han-Carlson Adder，Ladner-Fischer Adder），バランスのとれた加算器を構成する Conditional Sum Adder，Carry Select Adder，小面積な加算器を実現する Fixed-Block Carry Skip Adder，Variable-Block Carry Skip Adder を用いた．

本ジェネレータでは，以上の基数，アーキテクチャおよび演算アルゴリズムの組み合わせることで計 840 種類の RSA 暗号プロセッサを生成することができる．

## 5. 性能評価

本章では，設計パラメータを網羅的に変化させた性能評価実験により，提案ジェネレータから生成されるプロセッサの多様性と個々の設計指標に対する最適設計の性能を評価した．実験では，まず，提案するジェネレータによりすべての RSA 暗号プロセッサの Verilog HDL コードを生成した．そのうえで，その Verilog HDL コードを Design Compiler と Astro を用いて論理合成・配置配線し，配線を含めた遅延時間，回路面積，面積遅延積（遅延時間と回路面積の積），消費電力を見積もった．評価に使用したライブラリは，ST Microelectronics 社 90 nm CMOS スタンダードセルライブラリ[14] である．このとき，定格電圧 1.2 V，最悪条件下（電源電圧 1.08 V，125°C）とした．**図 6** は RSA 暗号プロセッサのレイアウト例である．以下では，設計パラメータごとの評価結果および得られたプロセッサの性能により提案ジェネレータの有効性を示し，RSA 暗号プロセッサの系統的設計について考察する．

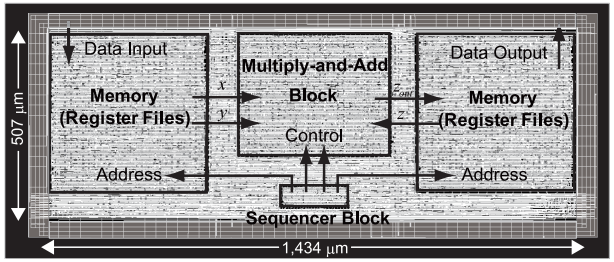
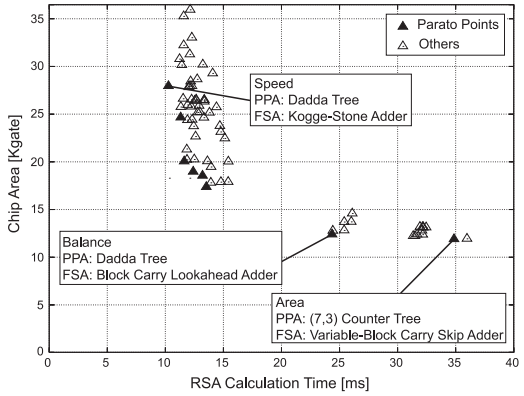
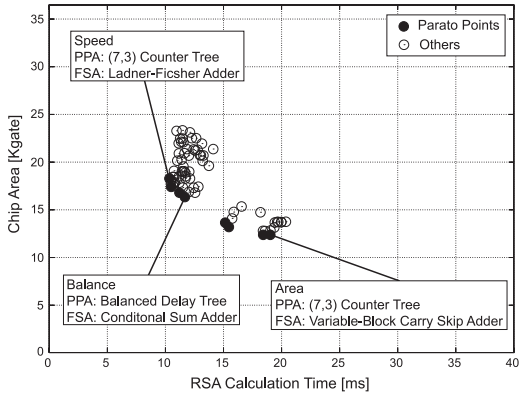



**図 6**　配置配線後のレイアウト例
Fig. 6　Post P&R layout.

まず，アーキテクチャおよび演算器のアルゴリズムの影響を示すため，基数 $2^{32}$ 時に生成可能な全 84 種類の性能分布を**図 7** に示す．ここで，横軸は 1,024 ビット RSA 暗号の計算




(a) Type-I




(b) Type-II

**図 7**　RSA 暗号プロセッサの性能分布（基数 $2^{32}$）
Fig. 7　Performance of RSA processors (Radix $2^{32}$).



© 2010 Information Processing Society of Japan

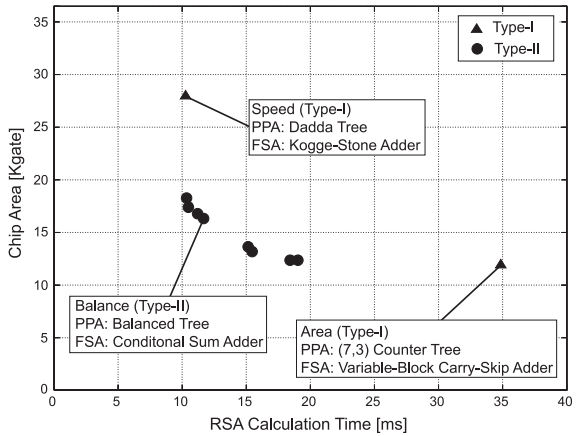



**図 8**　パレートフロンティア（基数 $2^{32}$）
Fig. 8　Pareto frontier (Radix $2^{32}$).

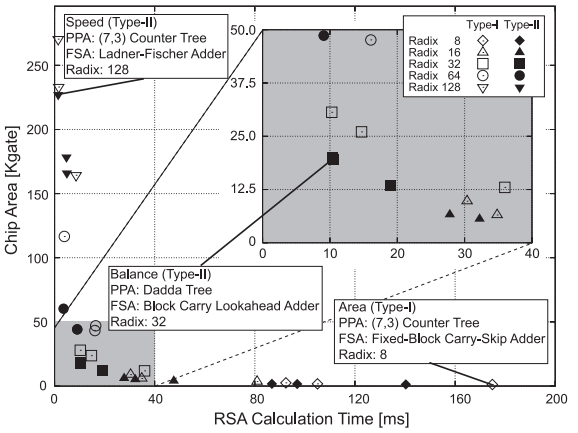



**図 9**　様々な基数での最高性能
Fig. 9　Best performance for various radices.

時間，縦軸は回路面積を表し，左下に分布するほど高い性能を示す．回路面積にはメモリおよび制御ブロックは含めず，演算ブロックのみで見積もった．この図より，アーキテクチャのタイプおよび積和演算器のアルゴリズムによって，速度にして 10.27〜35.97 [ms]，面積にして 11.8〜36.0 [Kgate] 程度の性能差が得られることを確認できる．

アーキテクチャ別では，Type-I からは全体的に幅広い性能分布が得られた．レジスタ数が少ないため，小面積な PPA および FSA アルゴリズムを選択した場合に回路面積に優れた結果が得られた．また，サイクル数を Type-II に比べ 2%少なくできるため，回路遅延が小さなアルゴリズムを選択した場合に処理時間に優れた結果が得られた．具体的には，(7,3) Counter Tree および Variable-Block Carry-Skip Adder を用いた場合に最小の回路面積 11.8 [Kgate] が実現され，Dadda Tree および Kogge-Stone Adder を用いた場合に最速の処理時間 10.27 [ms] が実現された．一方，Type-II からは Type-I と比べて密集した性能分布が得られた．図 7 (b) では，面積およそ 23.2 [Kgate]，速度 20.38 [ms] 以内に全プロセッサの性能が収まっており，回路効率に優れた構造が多く得られていることを確認できる．具体的には，Dadda Tree および Carry Select Adder を組み合わせた場合に最も高い回路効率 211.38 [ms·Kgate] が実現された．**図 8** にアーキテクチャと演算器のアルゴリズムの双方を考慮したパレートフロンティアを示す．2 種類のアーキテクチャと複数の算術アルゴリズムを備えることで，提案ジェネレータから連続的なパレート解が得られることが分かる．

次に，図 7 の結果から処理時間（Speed），回路面積（Area），回路効率（Balance）に優れた 3 種類の回路を各アーキテクチャで選択し，基数による性能分布を評価した．基数 $2^8$，$2^{16}$，$2^{32}$，$2^{64}$，$2^{128}$ における最高性能を**図 9** に示す．ここで，横軸は処理時間，縦軸は回路面積である．図より，基数が大きいほど高速動作，基数が小さいほど小面積となることが確認できる．また，図 7 と比較しても，性能が大きく変化しており，アーキテクチャや演算器のアルゴリズム以上に基数が大きな影響を与えることが分かる．

上記の結果を**表 2** にまとめる．ここで，MM Time はモンゴメリ乗算の処理時間，AD は面積遅延積である．Delay（DC）および Delay（Astro）は，それぞれ Design Compiler の論理合成時におけるゲート遅延のみのクリティカルパス，Astro の配置配線後のクリティカルパスを表す．消費電力は，セル自体の消費電力と配線での消費電力の合計で見積もった．太字で示す部分は，回路面積（Area），計算時間（Speed），回路効率（Balance）に最も優れた設計を表す．**表 3** に提案手法による最も優れた設計と従来設計との比較を示す．実装テクノロジが異なるため単純な比較は難しいが，文献 2)，3)，5) での回路のゲート数と演算サイクル数等も含め，可能な限り条件を揃えたうえで比較した．文献 2) の設計は，提案手法と同じく積和演算器ベースのアーキテクチャであるため，サイクル数はほぼ同一である．しかし，同じ基数 $2^{64}$ の設計で比べると，ゲート換算での回路面積および動作速度の面から提案する RSA 暗号プロセッサが優位であることが分かる．また，基数 2 の従来手法 3)，5)


© 2010 Information Processing Society of Japan

**表 2**　RSA 暗号プロセッサの性能比較
Table 2　Performance comparison of RSA cryptographic processors.

| Radix | Design | RSA Cycle | Delay (DC) [ns] | Delay (Astro) [ns] | MM Time [$\mu$s] | RSA Calc. Time [ms] | Chip Area [Kgate] | AD [ms·Kgate] | Power [mW] |
|---|---|---|---|---|---|---|---|---|---|
| **Type-I** | | | | | | | | | |
| $2^8$ | Speed | 51,286 K | 1.70 | 1.81 | 60.11 | 92.39 | 2.54 | 234.51 | 0.96 |
| | Balance | | 1.99 | 2.05 | 68.32 | 105.01 | 1.74 | 183.13 | 0.70 |
| | **Area** | | **3.09** | **3.42** | **113.78** | **174.89** | **1.25** | **218.51** | **0.34** |
| $2^{16}$ | Speed | 13,053 K | 2.10 | 2.34 | 19.76 | 30.37 | 8.90 | 270.31 | 2.69 |
| | Balance | | 2.47 | 2.68 | 22.66 | 34.83 | 5.95 | 207.08 | 1.77 |
| | Area | | 5.08 | 6.23 | 52.63 | 80.89 | 3.48 | 281.52 | 0.59 |
| $2^{32}$ | Speed | 3,381 K | 2.55 | 3.07 | 6.68 | 10.27 | 27.90 | 286.46 | 7.53 |
| | Balance | | 3.04 | 3.47 | 7.56 | 11.62 | 20.01 | 232.58 | 4.77 |
| | Area | | 9.73 | 10.75 | 23.39 | 35.97 | 11.85 | 426.09 | 1.38 |
| $2^{64}$ | Speed | 905 K | 2.95 | 4.34 | 2.50 | 3.85 | 116.58 | 449.04 | 30.37 |
| | Balance | | 4.74 | 5.77 | 3.33 | 5.13 | 52.40 | 268.60 | 9.04 |
| | Area | | 18.56 | 18.10 | 10.44 | 16.07 | 43.38 | 697.11 | 4.20 |
| $2^{128}$ | Speed | 258 K | 4.19 | 5.85 | 0.94 | 1.46 | 270.45 | 393.72 | 59.16 |
| | Balance | | 4.68 | 6.62 | 1.07 | 1.65 | 233.12 | 384.14 | 45.78 |
| | Area | | 36.42 | 34.57 | 5.57 | 8.60 | 164.39 | 1413.70 | 15.02 |
| **Type-II** | | | | | | | | | |
| $2^8$ | Speed | 51,483 K | 1.75 | 1.70 | 56.47 | 86.80 | 1.74 | 151.37 | 0.81 |
| | Balance | | 1.75 | 1.70 | 56.47 | 86.80 | 1.74 | 151.37 | 0.81 |
| | Area | | 2.38 | 2.74 | 91.29 | 140.32 | 1.32 | 184.64 | 0.51 |
| $2^{16}$ | Speed | 13,152 K | 2.01 | 2.14 | 18.08 | 27.80 | 6.02 | 167.28 | 2.03 |
| | Balance | | 2.22 | 2.48 | 20.96 | 32.23 | 5.07 | 163.49 | 1.66 |
| | Area | | 3.47 | 3.67 | 30.99 | 47.63 | 3.90 | 185.91 | 0.89 |
| $2^{32}$ | Speed | 3,430 K | 2.68 | 3.09 | 6.73 | 10.34 | 18.25 | 188.79 | 4.69 |
| | **Balance** | | **2.65** | **3.13** | **6.82** | **10.48** | **17.82** | **186.84** | **4.55** |
| | Area | | 5.51 | 5.68 | 12.38 | 19.03 | 12.35 | 234.94 | 1.87 |
| $2^{64}$ | Speed | 930 K | 3.46 | 3.99 | 2.30 | 3.54 | 60.40 | 213.79 | 12.29 |
| | Balance | | 3.46 | 3.99 | 2.30 | 3.54 | 60.40 | 213.79 | 12.29 |
| | Area | | 1.73 | 10.21 | 5.89 | 9.07 | 44.33 | 401.88 | 4.95 |
| $2^{128}$ | **Speed** | 270 K | **4.27** | **5.82** | **0.94** | **1.45** | **227.16** | **328.87** | **41.88** |
| | Balance | | 4.46 | 6.30 | 1.01 | 1.57 | 198.39 | 310.94 | 31.60 |
| | Area | | 19.21 | 19.75 | 3.18 | 4.91 | 166.08 | 815.93 | 16.07 |

と比べても，サイクル数の削減により計算時間が大幅に改善されている．結果として，提案ジェネレータにより，従来と比較しても数〜数十倍程度高性能なプロセッサが得られることが分かる．

以上のように，提案するジェネレータでは，パラメータの組合せにより小面積から高速演算まで多様な RSA 暗号プロセッサを生成することができる．特に，各パラメータは独立に選択可能であるため，一意に決まらないパラメータを設定の範囲内で適宜変更することで，


© 2010 Information Processing Society of Japan

**表 3**　従来手法との性能比較
Table 3　Performance comparison with conventional designs.

| Ref. | Tech. | Design | Radix | Delay [ns] | MM Cycle | MM Time [$\mu$s] | Gate [Kgate] |
|---|---|---|---|---|---|---|---|
| This Work | 90 nm CMOS | Speed (Type-II) | $2^{128}$ | 4.27 | 169 | 0.72 | 159.35 |
| | | Balance (Type-II) | $2^{32}$ | 2.65 | 2,209 | 5.85 | 12.64 |
| | | Area (Type-I) | $2^{8}$ | 3.09 | 33,281 | 102.84 | 0.88 |
| | | Balance (Type-II) | $2^{64}$ | 3.46 | 593 | 2.05 | 31.20 |
| 2) | 0.13 $\mu$m CMOS | 64 bit-Mult. | $2^{64}$ | 7.26 | 577 | 4.57 | 96.22 |
| 3) | 0.5 $\mu$m CMOS | 40 PEs×8 bit | 2 | 12.5 | 3,440 | 43 | 28 |
| 5) | 1.2 $\mu$m CMOS | 7 PEs×32 bit | 2 | 12.5 | 4,580 | 61 | 15 |

より高性能なプロセッサを得ることができる．たとえば，基数が用途や連携するシステムによりあらかじめ決定された場合，アーキテクチャと算術アルゴリズムの組合せから最良の設計を探索できる．また，提案ジェネレータは，基数や算術アルゴリズムの種類を容易に追加・変更することも可能であり，実装するデバイス（FPGA 等）やライブラリの特性に応じた最適設計も期待できる．一方で，本実験のような網羅的な合成を利用することで，提案ジェネレータを用いた系統的な設計も可能になると考えられる．最適なパラメータの組合せは実装するデバイスや使用するライブラリに応じて異なるが，今後類似のライブラリを用いた設計を行う場合にも，上記のパレートフロンティアおよびそのパラメータの組合せから効率的に最適な組合せを選択できると考えられる．

## 6. むすび

本稿では，高基数モンゴメリ乗算に基づく RSA 暗号プロセッサを 3 つの設計パラメータ（基数，アーキテクチャ，算術アルゴリズム）に応じて自動生成するジェネレータを提案した．本ジェネレータは，新たに考案したスケーラブルな 2 種類のアーキテクチャ（Type-I および Type-II）に加えて，5 種類の基数（$2^8$〜$2^{128}$）および 84 種類の演算器（PPA 7 種類，FSA 12 種類）を組み合わせることにより 840 種類の RSA 暗号プロセッサを生成することができる．生成可能なすべてのプロセッサを 90 nm CMOS スタンダードセルライブラリを用いて合成した結果，1,024 ビットの RSA 暗号に対して，1.25 [Kgate]@174.89 [ms] の小型実装（基数 $2^8$）から 227.16 [Kgate]@1.45 [ms] の高速実装（基数 $2^{128}$）まで，従来を上回る高い性能が得られた．また，各パラメータを網羅的に変更することで，従来にない幅広い性能を得られるとともに，その性能分布からパレート解を連続的に得られることを示した．以上の結果から，提案手法により用途に応じた適切な RSA 暗号プロセッサを設計できると考えられる．今後は，最適なパラメータの組合せを予測する手法を検討することが重要となる．また，ジェネレータを Web 上で公開するとともに，異なる CMOS テクノロジライブラリおよび FPGA による詳細な性能比較を行う予定である．

## 参考文献


1) Montgomery, P.L.: Modular multiplication without trial division, *Mathematics of Computation*, Vol.44, No.170, pp.519–521 (Apr. 1985).
2) Satoh, A. and Takano, K.: A scalable dual-field elliptic curve cryptographic processor, *IEEE Trans. Comput.*, Vol.52, No.4, pp.449–460 (2003).
3) Tenca, A.F. and Koc, C.K.: A scalable architecture for modular multiplication based on montgomery's algorithm, *IEEE Trans. Comput.*, Vol.52, No.9, pp.1215–1221 (2003).
4) Tenca, A.F., Todorov, G. and Koc, C.K.: High-radix design of a scalable modular multiplier, *CHES '01: Proc. 3rd International Workshop on Cryptographic Hardware and Embedded Systems*, pp.185–201, Springer-Verlag (2001).
5) Savas, E., Tenca, A.F. and Koc, C.K.: A scalable and unified multiplier architecture for finite fields GF(p) and GF($2^m$), *CHES '00: Proc. 2nd International Workshop on Cryptographic Hardware and Embedded Systems*, pp.277–292, Springer-Verlag (Aug. 2000).
6) Harris, D., Krishnamurthy, R., Mathew, S. and Hsu, S.: An improved unified scalable radix-2 montgomery multiplier, *ARITH '05: Proc. 17th IEEE Symposium on Computer Arithmetic*, pp.172–178, IEEE Computer Society (Sep. 2005).
7) Blum, T. and Paar, C.: Montgomery modular exponentiation on reconfigurable hardware, *ARITH '99: Proc. 14th IEEE Symposium on Computer Arithmetic*, pp.70–78, IEEE Computer Society (Apr. 1999).
8) Blum, T. and Paar, C.: High-radix montgomery modular exponentiation on reconfigurable hardware, *IEEE Trans. Comput.*, Vol.50, No.7, pp.759–764 (2001).
9) Koc, C.K., Acar, T. and Kaliski, B.S.: Analyzing and comparing montgomery multiplication algorithms, *IEEE Micro*, Vol.16, No.3, pp. 26–33 (1996).
10) Watanabe, Y., Homma, N., Aoki, T. and Higuchi, T.: Arithmetic module generator with algorithm optimization capability, *the 2008 IEEE International Symposium on Circuits and Systems*, pp.1796–1799 (May 2008).
11) Arithmetic ModuleGenerator basedon ARITH. http://www.aoki.ecei.tohoku.ac.jp/arith/mg/
12) Homma, N., Watanabe, Y., Aoki, T. and Higuchi, T.: Formal design of arithmetic circuits based on arithmetic description language, *IEICE Trans. Fundamentals.*, Vol.E89-A, No.12, pp.3500–3509 (2006).



© 2010 Information Processing Society of Japan

13) Koren, I.: *Computer arithmetic algorithms*, *2nd Edition*, A K Peters (2001).
14) Circuits Multi-Projets (CMP) CMOS 90 nm from STMicroelectronics. http://cmp.imag.fr/products/ic/?p=STCMOS090
15) Quisquater, J.J. and Couvreur, C.: Fast decipherment algorithm for RSA public-key cryptosystem, *Electronics Letters*, Vol.18, No.21, pp.905–907 (1982).

(平成 21 年 10 月 4 日受付)
(平成 22 年 6 月 3 日採録)

## 推　薦　文

本論文では，RSA 暗号プロセッサの自動生成手法を提案している．提案手法では，設計空間を効率的に探索することが可能となっており，実験結果によってもその有用性が示されている．論文としても完成度が高い．

（FIT2009 第 8 回情報科学技術フォーラム　プログラム委員会委員長　本位田真一）

**馬場　祐一**（正会員）

2008 年東北大学工学部情報工学科卒業．2010 年同大学大学院情報科学研究科修士課程修了．同年ルネサスエレクトロニクス（株）入社．現在に至る．暗号ハードウェアに関する研究に従事．

**宮本　篤志**

2005 年東北大学工学部情報工学科卒業．2007 年同大学大学院情報科学研究科修士課程修了．2010 年同博士課程修了．同年（株）日立製作所中央研究所入所．現在に至る．暗号ハードウェアに関する研究に従事．IEEE 会員．博士（情報科学）．

**本間　尚文**（正会員）

1997 年東北大学工学部情報工学科卒業．2001 年同大学大学院情報科学研究科博士課程修了．同年同研究科助手，2007 年同助教．2009 年同准教授，現在に至る．2002〜2006 年科学技術振興機構さきがけ研究者を兼任．2009〜2010 年 Telecom ParisTech 客員研究員．ハードウェアアルゴリズム，VLSI 設計技術，ハードウェアセキュリティに関する研究に従事．CRYPTREC 暗号実装委員会委員．IEEE，電子情報通信学会各会員．博士（情報科学）．

**青木　孝文**（正会員）

1988 年東北大学工学部電子工学科卒業．1992 年同大学大学院工学研究科博士課程修了．同年同大学工学部助手，1994 年同大学大学院情報科学研究科助手，1996 年同助教授，2002 年同教授，現在に至る．超高速ディジタル計算の理論，画像センシング，映像信号処理，バイオメトリクス，VLSI 設計技術，分子コンピューティングに関する研究に従事．英国電気学会フレミング賞およびマウントバッテン賞ほかを受賞．IEEE，計測自動制御学会，電子情報通信学会各会員．博士（工学）．

**佐藤　　証**

1989 年早稲田大学大学院理工学研究科電気工学専攻修士課程修了．同年日本アイ・ビー・エム（株）東京基礎研究所入所．1999 年早稲田大学より博士（工学）授与．2007 年（独）産業技術総合研究所入所．現在に至る．情報セキュリティに関するアルゴリズムおよび，その高性能 VLSI 実装方式に関する研究に従事．博士（工学）．



© 2010 Information Processing Society of Japan