素因数分解ハードウェアの研究・開発動向について

伊豆哲也[†] 國廣昇[††] 下山武司[†]

素因数分解の高速化を目的とした専用ハードウェアの実現可能性が活発に議論されている．理論的な研究は急速に進展しているが，さまざまな前提を用いていることから，実際の脅威評価が困難な状況にある．本稿は，最速の素因数分解法である数体篩法の実現を目的とした専用ハードウェアの研究・開発状況を整理するとともに，現時点での脅威について考察する．

On the Dedicated Factoring Hardware

Tetsuya Izu,[†] Noboru Kunihiro[††] and Takeshi Shimoyama[†]

Dedicated factoring devices have attracted much attention in recent years. While a large number of theoretic results are shown, it is hard to evaluate since various assumptions are required. This paper surveys dedicated factoring hardware of the Number Field Sieve method, the fastest integer factorization algorithm. Also, a current threat of these devices are considered.

1. はじめに

公開鍵暗号の標準である RSA 暗号の安全性は，巨大合成数の素因数分解が難しいという事実に依存する．したがって，素因数分解は RSA 暗号に対する攻撃を意味し，Lenstra らによって 1990 年に提案された最良の素因数分解法である数体篩法（Number Field Sieve method，NFS）[36] の 2007 年 2 月時点の分解記録が 663 ビット[4] であることから，現在の RSA 暗号が公開鍵として標準的に使用する 1,024 ビット合成数の分解は当面は困難と考えられている．しかし，以前に米標準共通鍵ブロック暗号であった DES の解読において専用ハードウェア（DES CRACKER）が劇的な時間短縮を可能にしたように，素因数分解専用ハードウェアを RSA 暗号にとっての潜在的な脅威と考えることは自然であり，その評価が急務となっている．特に，専用装置によって 1,024 ビット合成数が可能となった場合，RSA 暗号を利用したシステムへの影響は甚大であろうことから，1,024 ビット合成数を分解可能な専用装置の製造可能性・動作可能性の議論は重要なテーマといえる．

本稿は，専用装置の製造可能性・動作可能性を議論するための礎として，これまでに提案されている素因数分解専用装置のデザイン例・実装例に関する研究・開発動向を報告する．数体篩法は多項式選択，関係式探索，線形代数，平方根計算の 4 つのステップから構成されるが，理論的にも実験的にも計算時間の大半を占めるのは関係式探索ステップと線形代数ステップであり，素因数分解専用装置ハードウェアのターゲットともなっている．本稿では，関係式探索ステップ（篩処理部，関係式検査部）と線形代数ステップを対象としたデザイン例と実装例を扱う．またこれら報告内容をもとに，現時点における素因数分解ハードウェア装置の脅威について考察する．

本稿の構成は以下のとおりである．2 章で数体篩法の関係式探索ステップと線形代数ステップで必要となる処理内容を説明する．次にこれらステップに対するハードウェアデザイン例を 3 章で，実装例を 4 章で紹介する．最後に 5 章において考察を行う．

2. 数体篩法

本章では数体篩法の関係式探索ステップと線形代数ステップで必要となる処理内容を簡単に説明する．アルゴリズム全体の処理内容については既存文献[29),35)] を参照されたい．

† 富士通株式会社
FUJITSU Limited
†† 電気通信大学
The University of Electro-Communications

本研究成果の一部は，独立行政法人情報通信機構（NICT）の委託研究「素因数分解の困難性に基づく技術評価に関する研究開発」の支援のもとで行われた．

2.1 関係式探索ステップ

整数ペア (a,b) が以下の 3 条件を満たすとき，ペア (a,b) を関係（relation）と呼ぶ：

- $\gcd(a,b)=1$
- $F_{\mathrm{r}}(a,b)$ は B_{r}-smooth．ただし 1 次の整数係数 2 変数多項式 $F_{\mathrm{r}}(x,y)=mx+y$，自然数 B_{r} は与えられたパラメータ．
- $F_{\mathrm{a}}(a,b)$ は B_{a}-smooth．ただし d 次の整数係数 2 変数多項式 $F_{\mathrm{a}}(x,y)=(-x)^d f_{\mathrm{a}}(-y/x)$，$d$ 次の既約な整数係数 1 変数多項式 $f_{\mathrm{a}}(z)=c_d z^d+\cdots+c_0$，自然数 B_{a} は与えられたパラメータ．

ここで自然数 F が自然数 B に対して B-smooth であるとは，F が B 以下の素因数の積に分解できることをいう．2 つ目の条件を有理的（rational），3 つ目の条件を代数的（algebraic）と形容する．これらの条件は類似しているため，以下，多項式 $F(x,y)$ は $F_{\mathrm{r}}(x,y)$ および $F_{\mathrm{a}}(x,y)$ を，自然数 B は B_{r} および B_{a} を表すことにする．関係式探索ステップの目標は，与えられた 2 次元領域 $\{\,(a,b)\in\mathbb{Z}\times\mathbb{N}\mid -H_a\le a\le H_a,\ 1\le b\le H_b\,\}$ から，一定個数以上の関係 (a,b) を見つけることである．見つける関係は一定の個数以上ならばどのような組合せでもよく，すべてを見つける必要もない．

巨大整数の分解で用いる 2 次元領域は巨大☆なため，部分領域に分割して処理を行う（たとえば，ある b に対して $F(a,b)$ が B-smooth となる a を $\{-H_a,\ldots,H_a\}$ から探索する）．B-smooth であるかを判定するのに B 以下の素数で順番に割っていく方法が考えられるが，多数回の整数除算が必要となるため効率が悪い．そこで $F(a,b)=\prod_{p_i\mid F(a,b),\ p_i\le B} p_i^{e_i}$ のときに成立する

$$\log F(a,b)=\sum e_i\log p_i\approx\sum\lfloor\log p_i\rceil$$

という近似式を利用する（$\lfloor x\rceil$ は実数 x に最も近い整数を表す）．このような近似が可能なのは，ほとんどの大きな素因数 p_i について $p_i\mid F(a,b)$ ならば $e_i=1$ となること，メモリ効率の観点から対数の値を整数に丸めた方が好ましいからである．そこで探索する際に各 (a,b) に対応する変数 $s(a,b)$（初期値 0）を用意し，整数 $F(a,b)$ が素数 p_i で割り切れるときに $\lfloor\log p_i\rceil$ を $s(a,b)$ に加える．この操作を B 以下のすべての素数に対して行い，最終的な $s(a,b)$ の値が $\log F(a,b)$ に近い場合，(a,b) を関係の候補（candidate）として抽出する．“$\lfloor\log p_i\rceil$ を $s(a,b)$ に加える”という操作を効率化するため，以下の処理を用いる：

☆ たとえば 1,024 ビット合成数の分解では $H_a=5.5\times10^{14}$，$H_b=2.7\times10^8$ が必要とされている[39]．

多項式 $F(x,y)$ は，$F(a,b)$ が p_i で割り切れるならば，$F(a+p_i,b)$ も p_i で割り切れる，という性質を持つ．そこで $\lfloor\log p_i\rceil$ を加えるという操作を連続的に行うために，与えられたある b に対して，処理対象となる $(a,b)=(-H_a,b),\ (-H_a+1,b),\ldots,(H_a,b)$ に対応する変数 $s(-H_a),\ s(-H_a+1),\ldots,s(H_a)$（初期値 0）を用意する．$F(a,b)$ が p_i で割り切れる最小の a を a_i と書くと，上の性質から $F(a_i+p_i,b),\ F(a_i+p_i,b),\ldots$ も p_i で割り切れるため，変数 $s(a_i),\ s(a_i+p_i),\ldots$ に $\lfloor\log p_i\rceil$ を加えていく．この操作を B 以下のすべての素数について繰り返すことで，$F(a,b)$ が B-smooth となる a の候補を $-H_a\le a\le H_a$ の中から効率的に見つけることができる．このような操作を篩（sieve），ここまでの処理部を篩処理部と呼ぶ．上で説明した篩は線篩（line sieve）と呼ばれる手法であるが，現在のソフトウェア実装ではより効率的な格子篩（lattice sieve）が使用されている[4]．本稿では格子篩の説明を割愛するが，本質的に必要となる処理内容は線篩と大差ない．

篩処理部の後，得られた候補 (a,b) が関係になっているかどうか，つまり $F(a,b)$ が B-smooth になっているかを判定する必要がある．この判定を行うのが関係式検査部であり，具体的には前処理（自明な素因数の除去）・ミニ合成数判定・ミニ素因数分解から構成される．前処理では $F(a,b)$ から B 以下の素数を除去し，余因子 $R=F(a,b)/\prod p_i^{e_i}$ を求める．前処理は B 以下の素数による試行除算で実現できるが，篩処理部で $\lfloor\log p_i\rceil$ を加える際に p_i の値も同時に保管することで，前処理の大部分を省略することも可能である．R が 1 でなければ $F(a,b)$ は B-smooth でないため，(a,b) を棄却することになるが，R が B と数桁程度しか違わない素数の場合，あるいは B と数桁程度しか違わない素数の積である場合，この (a,b) も関係に含めて考える（Large Prime Variation，LPV）☆☆．そこでミニ合成数判定では R が合成数かを判定し，合成数ならば次のミニ素因数分解に進む．そうでなければ R を素数と見なし，$R\le B'$ ならば (a,b) を関係として抽出する．なおミニ合成数判定では，低い確率ならば合成数を擬素数と判定してもよい．ミニ素因数分解では，合成数 R を分解し，その素因数がすべて B' 以下ならば (a,b) を関係として抽出する．ミニ素因数分解でも，低い確率ならば分解に失敗してよい．

☆☆ さらには $F(a,b)=RR'\prod p_i^{e_i}$（$R,R'\le B'$）のような (a,b) も用いることがある（2 LPV）．巨大整数の数体篩法による素因数分解では，有理側に 2 LPV，代数側に 3 LPV を用いることがある（2-3 LPV）．

ミニ素因数分解には，従来は ρ 法や SQUFOF 法が用いられていたが[1]，近年では複数多項式 2 次篩法が用いられている[32]．またハードウェアでは楕円曲線法が用いられている[15)~17),40)]．

2.2 線形代数ステップ

線形代数ステップの目標は GF(2) 上の $D \times D$ の行列 $\tilde{A}$ と D 次のベクトル v に対して，$\tilde{A}v = 0$ の（自明でない）解を GF(2) 上で求めることである．効率的な手法である Wiedemann アルゴリズムは，$\tilde{A}$ から定まる GF(2) 上の $D \times D$ の行列 A に対して

$$A^k v \quad (k = 1, 2, \ldots, 2D)$$

を計算することで解を求めるため，行列とベクトルの積 Av の効率的処理が必要となる．ここで行列 A は疎（sparse），つまり行あたりの 1 の平均個数 h がサイズ D に比べて小さいという特徴を持つ☆．積 Av は

(1) 積の計算：各 i, j に対して $a_{i,j}v_i$ を求める
(2) 和の計算：各 i に対して $\sum_j a_{i,j}v_i$ を求める

の繰返しによって算出される．(1)，(2) はいずれも GF(2) で計算され，行列 A が疎であることから，(1) では $a_{i,j} = 1$ かつ $v_i = 1$ の項だけを，(2) では $a_{i,j}v_i = 1$ の項だけを処理する方が効率的である．したがって，行列 A, ベクトル v の値を保持する場合，すべての成分の値を保持するのではなく，要素値が 1 の $a_{i,j}$ と v_i のみを保持することになる．

3. デザイン例

本章では，これまでに提案された素因数分解ハードウェアの主要な提案デザイン例を紹介する．

3.1 関係式探索ステップ

3.1.1 TWINKLE（1999 年）

TWINKLE（The Weizemann INstitute Key Locating Engine）は，篩処理部を線篩に基づいて処理するデザインであり，1999 年に Shamir によって提案された[41]．ただし実験報告はなされていない．TWINKLE は発光部と受光部から構成され，発光部は B 以下の各素数に対応する多数の LED を有する．$F(a, b)$ が素数 p_i で割り切れる場合，素数 p_i に対応する LED_i は $\lfloor \log p_i \rceil$ に比例する光を放つ．LED_i の発行周期は p_i クロックごとであり，各 LED は独立に発光する（複数の LED が同時に発光することもある）．受光部は変数 $s(a, b)$ を表す光学素子であり，受光量の合計値が閾値を超えた (a, b) を候補として抽出する．TWINKLE は原理が簡単なため実現も容易であるが，費用が高価であり，並列化した場合に発光部と受光部をそれぞれ複数用意しなければならないという構造的な欠点を持つ．実際，TWINKLE の性能限界は 512 ビットであり，768 ビットの処理は不可能と評価されている[37]．

3.1.2 DSH（2003 年）

DSH（Dedicated Sieving Hardware）は，篩処理部を線篩に基づいて処理するデザインであり，2003 年に Geiselmann らによって提案された[18]．ただし実験報告はなされていない．DSH の性能限界は 512 ビットであり，4 日程度で処理可能と述べられている．DSH の特徴は，パケットの送信制御全体にソーティングを用いる点であるが，Bernstein が線形代数ステップ用デザインで用いたアイディア[6] を，関係式ステップに適用したと見なすことができる．

3.1.3 TWIRL（2003 年）

TWIRL（The Weizemann Institute Relation Locator）は，篩処理部を線篩に基づいて処理するデザインであり，2003 年に Shamir らによって提案された[42]．ただし実験報告はなされていない．約 10 億円の製造費（直径 300 mm の ASIC ウェハ 600 枚）を投資した場合，TWIRL は約 1 年で 1,024 ビット合成数の処理が可能と主張されている点が特徴的である（さらには 512 ビットや 756 ビットでも他のデザインに対する圧倒的な優位性が主張されている）．しかし代数的な TWIRL のチップサイズがウェハ 1 枚分の面積を占めるなど，TWIRL の実現・運用の可能性についてはさまざまな指摘がなされている．たとえば CRYPTREC による評価では，(1,024 ビットの場合における）否定的な分析結果が報告されている[11),27)]．

TWIRL は TWINKLE を ASIC 上で実現させた構造を有する．ただし 1 組のエミッタ（発光部）に対して多数組の積算部（受光部）が設置され，積算部どうしでもデータ転送が可能となっている．エミッタは $\lfloor \log p \rceil$ とこの値を蓄積すべき積算部のアドレスをパケットと呼ばれるデータとして送信し，積算部は受け取ったパケットを蓄積する．回路内のパケット占有率を高めるために，TWIRL は素数の大きさに応じた 3 種類のエミッタを用いる．しかし大量のパケットをそれぞれのターゲット積算部に送信するために巨大なソータが必要となり，TWIRL チップの巨大化の原因となっている．

なお TWIRL では，$F(a, b)$ が p_i で割り切れる際に，“$F(a, b)$ は p_i で割り切れる” という情報（diary）をパケットとは別に発行・制御することとなっており（しかし関係式検査部は TWIRL の外部で処理される），

☆ たとえば 512 ビット合成数の素因数分解実験では $D \approx 6700000$, $h = 63$ となることが報告されている[7]．

その制御の困難性も指摘されている（3.2.1 項参照）．

3.1.4 YASD（2004 年）

YASD（Yet Another Sieving Device）は，篩処理部を線篩に基づいて処理するデザインであり，2004 年に Geiselmann らによって提案された[20),21)]．ただし実験報告はなされていない．YASD の限界性能は 768 ビットであり，約 600 日で処理可能と述べられている．これは TWIRL（の 768 ビット版）に比べ，同じ製造費用を投資した場合に，約 6.3 倍遅い結果となっている．YASD を 1,024 ビットに拡張させた場合の性能評価[24)~26)] では，約 34 億円の製造費を投資した場合，YASD は約 1 年で処理可能と見積もられている．

YASD の特徴は，パケットの送信制御全体に（ソーティングよりも効率に優れた）ルーティングを用いる点であるが，Lenstra らが線形代数ステップ用デザインで用いたアイディア[38)] を，関係式ステップに適用したと見なすことができる．しかしパケット制御をルーティングというブラックボックスに任せているため，効率の低下を招いている．

なお TWIRL の diary に相当する情報を YASD では footnote として保管するが，関係式検査部の外部処理が想定されている．

3.1.5 SHARK（2005 年）

SHARK は，関係式探索ステップを格子篩に基づいて処理するデザインであり，2005 年に Franke らによって提案された[9),10)]．ただし実験報告はなされていない．SHARK も ASIC 実装を前提としており，YASD と同様に 1,024 ビットの処理が可能であることが主張されている．見積りでは，約 92 億円の製造費を投資した場合，約 1 年の計算時間が必要とされている（このほかにも電力供給機・消費電力などの見積りも提示されている）．

SHARK は TWIRL と同様の構成をとるが，ソータの巨大化を防ぐためにバケツソート[2),3)] を用いている．バケツソートとはデータを 2 段階に分けてソートする方法で，SHARK では 1 回目のソートに Butterfly ソートを用いる．TWIRL の巨大ソータに比べ，これらソータの実現性は高まるが，効率も低下するため製造費の膨張を招いている．SHARK のもう 1 つの特徴は，関係式検査部もデバイス内で処理する点であり，ミニ素因数分解用に楕円曲線法を処理するコプロセッサの利用を想定している．

3.1.6 TWIRL の改良（2006 年）

TWIRL では diary の制御に巨大な回路が必要となるという問題があった．そこで Geiseilmann らは diary を保管する代わりに，関係式検査部に専用ハードウェアを用いるという改良法を 2006 年に提案した[16)]．彼らのアイディアは SHARK のアイディアを推し進めたもので，前処理・ミニ素数判定・ミニ素因数分解のすべてを専用ハードウェアが処理することを想定している．専用ハードウェアには Franke らの楕円曲線法（ECM）ユニット[8),40),46)] が用いられ，6 個の ECM ユニットを 1 組の ECM クラスタと見なし，面積と入力値の大きさに応じた個数の ECM クラスタを TWIRL に搭載する．このクラスタのチップサイズは Pentium 4 と同程度で，1,024 ビットの場合には約 30 個の ECM クラスタを用いることで，TWIRL の実現可能性の向上が主張されている．なお ECM ユニットのコスト・処理時間の全体に占める割合は小さいため，TWIRL 自体のコスト・処理時間には影響を与えないとされている．なお本改良では ECM ユニットをブラックボックスとして使用しているため，Franke らのデザインを Gaj らのデザイン[14),15)] に変更することは容易であり，（理論的な）専有面積のさらなる削減が可能であろう．

3.1.7 TWIRL と YASD の融合（2006 年）

TWIRL のもう 1 つの問題である巨大ソータを回避するために，Geiselmann らはソータ部分を YASD のようなルーティング回路に置き換える改良法を 2006 年に提案した[22),23)]．TWIRL に比べて処理速度が約 1/3 になるものの，ASIC でのチップサイズが約 493 mm^2 におさえることができるため，実現可能性の向上が主張されている．

3.1.8 比　　較

関係式探索ステップ（篩処理部）に対する主要なハードウェアデザイン例の比較として，TWIRL[42)]，YASD[20),21)]，SHARK[9),10)] の回路規模・処理時間・費用を**表 1** にまとめる．ここでは提案者らによる見積り値を用いたが，1,024 ビットの場合の YASD に関しては，廣田らの見積り値を用いた[24)~26)]．また費用では製造コストだけを考慮しており，初期開発費用・人件費・歩留まり・電源回路などのコストは除外した．

3.2 線形代数ステップ

3.2.1 Bernstein 回路（2001 年）

Bernstein 回路とは線形代数ステップをソーティングに基づいて処理するデザインであり，2001 年に Bernstein によって提案された[6)]．ただし実験報告はなされていない．線形代数ステップのソフトウェア処理において，CPU はほとんどの時間で使用されているのに対しメモリのアクセス頻度は低いことが知られている．そこで Bernstein は，全メモリ領域を同時に（並列に）動作させることで計算時間の短縮につながる

表 1 関係式探索ステップ（篩処理部）のハードウェアデザインの比較
Table 1 A comparison of dedicated factoring devices for the sieving part.

		1,024 ビット	768 ビット
TWIRL	回路規模	15,960 mm^2 × 8 + 66,000 mm^2	1,330 mm^2 + 4,430 mm^2
（1 GHz）	時間/費用	194 年/15,000 ドル	2.3 年/750 ドル
SHARK	回路規模	1/4 wafer + DRAM	（評価なし）
（1 GHz）	時間/費用	2,300 年/40,000 ドル	
YASD	回路規模	42,200 mm^2	2,400 mm^2
（500 MHz）	時間/費用	10,652 年/3,200 ドル	34.5 年/250 ドル

表 2 Bernstein 回路のコスト見積り
Table 2 An evaluation of Bernstein circuit.

（小行列）

ハードウェア	ρ	K	ウェハ数/チップ数	コスト（ドル）	実行時間（秒）	スループットコスト
カスタム 1		1	273	4,100,000	1,210,000 （14 日）	4.96×10^{12}
カスタム 1		$\gg 1$	273	4,100,000	182,000 （50 時間）	7.44×10^{11}

表 3 Lenstra 回路のコスト見積り
Table 3 An evaluation of Lenstra circuit.

（小行列）

ハードウェア	ρ	K	ウェハ数/チップ数	コスト（ドル）	実行時間（秒）	スループットコスト
カスタム 1	0.51	107	19	94,600	1,440 （24 分）	1.36×10^8
	0.11	532	288	1,440,000	180 （3 分）	2.60×10^8
カスタム 2	42.10	208	1	5,000	21,900 （6.1 時間）	1.10×10^8
	216.16	42	0.37	2,500	341,000 （4 日）	8.53×10^8
FPGA	5,473.24	25	64	13,800	15,900,000 （184 日）	2.20×10^{11}
	243.35	60	2,500	380,000	1,420,000 （17 日）	5.40×10^{11}

（大行列）

ハードウェア	ρ	K	ウェハ数/チップ数	コスト（ドル）	実行時間（秒）	スループットコスト
カスタム 1	0.51	136	6,030	30.1 M	5.04×10^6 （58 日）	1.52×10^{14}
	0.11	663	9,000	500.0 M	6.40×10^5 （7.4 日）	2.88×10^{14}
カスタム 2	4112	306	391	2.0 M	6.87×10^7 （2.2 年）	1.34×10^{14}
	261.60	52	120	0.6 M	1.49×10^9 （47 年）	8.95×10^{14}
FPGA	17,757.70	99	13,567	3.5 M	3.44×10^9 （1088 年）	1.19×10^{17}
	144.41	471	6.6×10^6	1,000.0 M	1.14×10^9 （36 年）	1.13×10^{18}

と考え，ASIC 上での専用ハードウェアデザインを提案した．Bernstein 回路自体の効率は好ましいものではなかったが，彼の着想はそれ以降の素因数分解ハードウェアを設計するうえでの基本的な考え方となっており，現在の素因数分解ハードウェア研究が隆盛するきっかけと考えられている．Lenstra らによる評価[38]によると，1,024 ビット合成数の素因数分解で使用するある行列（“小行列”）に対し，約 41 億円の製造費を投資した場合，約 14 日で処理可能と見積もられている（3.2.3 項参照）．さらに Wiedemann アルゴリズムのパラメータを最適化させると，実行時間が約 50 時間にまで短縮可能であるとされている．

Bernstein 回路は与えられた $D \times D$ の行列 A と D 次のベクトル v の要素 1 の成分データを保持し，これらデータをソートすることで，行列とベクトルの積 Av を計算する．ほとんど移動することがないデータが多く存在すること，またソート終了直前にはほとんどのデータが移動しないことが効率を妨げる原因として指摘されている．

3.2.2 Lenstra 回路（2002 年）

Lenstra 回路とは線形代数ステップをルーティングに基づいて処理するデザインであり，Bernstein 回路の改良として 2002 年に Lenstra らによって提案された[38]．Bernstein 回路のソートにおける問題を克服するために，Lenstra 回路はルーティングを使用することで，データ移動を一定に保つことで行列積計算の効率化を実現している．Lenstra 回路を用いた場合，“小行列” に対しては約 50 万円の製造費と約 6.1 時間の実行時間で計算可能と見積もられている（3.2.3 項参照）．

3.2.3 比　　較

Lenstra らによる Bernstein 回路と Lenstra 回路の，1,024 ビット合成数の処理時に必要となる 2 種類の行列に対する性能比較を**表 2**，**表 3** に示す[38]．ここで

小行列のサイズは $D \approx 4 \times 10^7$，大行列のサイズは $D \approx 1.8 \times 10^{10}$ であり，いずれの場合も $h = 100$ とする．またハードウェアタイプのカスタム 1 は論理ウェハを，カスタム 2 は DRAM ウェハを想定し，FPGA には Altera Straix EP1S25F1020C7 のパラメータ（コスト 150 ドル）を用いている．なお ρ，K は Lenstra 回路のパラメータである．またスループットコストはコストと実行時間の積を表す．

3.2.4 Lenstra 回路の改良（2003 年）

Bernstein/Lenstra 回路を提案どおりに実装すると複数のウェハが必要となり，ウェハ間通信を考慮する必要がある．そこで回路を複数の LSI に分割実装する方法を 2003 年に Geiselmann らは提案した[19]．彼らのアイディアは，$D \times D$ の行列 A を $s \times s$ の小行列に分割し，個々に積を計算することである．1,024 ビット合成数の素因数分解に必要な行列計算でも，11.4 mm 角の LSI を 529 個使用することで計算可能と見積もられている．

4. 実 装 例

本章では，これまでに報告された素因数分解ハードウェアの実装例を紹介する．

4.1 関係式探索ステップ（篩処理部）

4.1.1 リコンフィグアブルプロセッサへの実装（2005 年）

2005 年に Shimoyama，Izu，Kogure は，関係式探索ステップの線篩に基づく篩処理部を IPFlex 社のリコンフィグアブルプロセッサ DAPDNA-2 に実装した結果を報告した[30),43),45)]．実験では 330 ビットの合成数（RSA100）の篩処理を部分的に行っているが，ソフトウェア実装に比べて約 40 倍遅いことが報告されている．

4.1.2 FPGA 実装（2006 年）

2006 年に富士通，富士通研究所，情報通信機構は関係式探索ステップを FPGA に実装した結果を報告した[12),13),31),44)]．篩処理部は線篩によって，関係式検査部は ρ 法によって処理された．実験では 432 ビットの合成数の篩処理が約 30 日で処理できたことが報告され，格子篩によるソフトウェア実装と同等の性能であることが主張されている．ただし製造費用は明らかにされていない．

4.2 関係式探索ステップ（関係式検査部）

4.2.1 Franke らによるデザイン（2005 年）

Franke らは楕円曲線法（ECM）によるミニ素因数分解ハードウェアのデザインを 2005 年に提案し，あわせて FPGA への実装結果を報告した[8),40),46)]．彼らの目的は楕円曲線法の忠実な実現であり，デザインにおける特段の特徴は見受けられない．実装では Xilinx 社の FPGA（Virtex 2000E-6）と ARM 社の RISC プロセッサ（ARM7TDMI）が組み込まれたシステム・オン・チップ（周波数 25 MHz）が用いられており，楕円曲線法の制御部は ARM7，楕円曲線演算は FPGA にて処理されている．FPGA にはロジック・メモリ・乗算器（1 個）・加算器（1 個）が搭載され，モンゴメリ乗算には Tenca らの手法[47)] が用いられている．Franke らのデザインは SHARK[10)] への組み込みを前提としており，それに応じたパラメータが使用されている．FPGA の占有比は 5.8%，メモリの占有比は 27%であり，1 個の FPGA に 3 つの楕円曲線演算器（ECM ユニット）が搭載可能となっている．

4.2.2 Gaj らによるデザイン（2006 年）

Gaj らは楕円曲線法のハードウェアデザインの提案と実装報告を 2006 年に行った[14),15)]．彼らの目的も関係式探索ステップハードへの組み込みであるが，Franke らのデザインを抜本的に見直すことで，数倍から十数倍の高速化を実現した．実装では Xilinx 社の FPGA（Virtex 2000E-6）が使用され，制御部も ECM ユニットも FPGA 内に搭載，制御部は複数の ECM ユニットを制御可能となっている．ECM ユニットにはロジック・メモリ・乗算器（2 個）・加算器（1 個）が搭載されているが，モンゴメリ乗算には McIvor らの手法[28)] が用いられている．高速化が実現できた理由として，Franke らとは異なる Montgomery 乗算アルゴリズムを用いたこと，乗算器を 2 個利用することで乗算器・加算器の占有率を高めたことがあげられている．このため ECM ユニットの占有比は 16%と高くなっているが，メモリの効率的管理を行うことでメモリの占有比を 1.3%に抑えられており，1 個の FPGA に 7 つの ECM ユニットが搭載可能となっている．

4.3 線形代数ステップ

4.3.1 Lenstra 回路の FPGA 実装（2004 年）

2004 年に Bajracharya らは，Xilinx Virtex II XC2V8000 という FPGA の RTL シミュレーションにおける線形代数ステップの実装報告を行った[5)]．報告では，$D = 2{,}304$ での行列とベクトルの 1 回の積計算に 18 ns（周波数 55.5 MHZ）を要するとされている．これは 1 個の FPGA で処理できる最大次元であるが，Geiselman らによる改良法[19)] を用いるとさらに大きな次元も処理可能であり，512 ビット合成数に対応する行列計算（$D = 6.7 \times 10^6$）の場合で約 600 日（FPGA 1 個の場合），1,024 ビット合成数に対応する行列計算（$D = 10^{10}$）の場合で約 1.9×10^{12} 日

（FPGA 1 個の場合）が必要であると試算されている．

5. 考　　察

素因数分解専用ハードウェア装置の脅威を想定する場合，デザインが示されたことをもって脅威と見なすのは 1 つの考え方である．この場合，1,024 ビット合成数の分解が可能な関係式探索ステップと線形代数ステップのハードウェアデザインが示されていることから，1,024 ビット合成数を用いた RSA 暗号は危険という結論を得る．

しかし，デザインを示すことと，そのデザインに基づいたデバイスを実装し，理論通りに動作させるまでに検討すべきことは多い（具体的なレイアウト・デバイスの詳細な仕様・製造可能性・開発費用・動作可能性など）．したがって素因数分解専用ハードウェアの脅威は，実機を用いた実験まで行われているという尺度を用いる方が現実的であろう．前章までの結果をこの観点から整理すると，1,024 ビット合成数を想定した場合，関係式検査部用のハードウェアは十分に製造・処理可能なレベルにあるといえる．他方で篩処理部用のハードウェアの実験では，最も単純なデザインが実装されたにとどまっており，すぐには 1,024 ビット合成数が処理できるような装置が（特に ASIC で）製造され，現実的な時間内に処理される可能性は低いといわざるをえない．

しかし前章までで紹介したように，素因数分解装置に関する研究は理論面・実装面の両面で着実に進歩しているため，継続的な調査・評価は必須であると考える．特に上記のような篩処理部と関係式検査部のアンバランスな状況に着目して，篩処理と検査処理の重み付けを再考し，たとえば検査処理の比重をより高くするような設計指針が考えられる．したがって，これまでのように篩処理，検査処理，線形代数処理を個別に評価するだけでなく，トータルとしての処理性能を考慮する必要があると考える．

6. ま と め

本稿は数体篩法の関係式探索ステップと線形代数ステップに対する素因数分解ハードウェアのデザイン例と実装例の研究・開発状況を報告した．理論面に比べて実装面での報告例が少ないことから，1,024 ビット合成数が現実的な費用・時間内で素因数分解される危険性は低いと考えるが，関連研究は理論面・実装面の両面で着実な進歩を見せており，継続的な調査・評価は必須であると考える．

参 考 文 献

1) 青木和麻呂，伊豆哲也，植田広樹，下山武司：一般数体篩法実装実験（2）— ミニ素因数分解・ミニ素数判定，電子情報通信学会技術研究報告，ISEC 2003-152, pp.229–234 (2004).
2) 青木和麻呂，植田弘樹：一般数体篩法実装実験（5）— バケツ整列を利用した篩の高速化，電子情報通信学会技術研究報告，ISEC 2004-25, pp.77–81 (2004).
3) Aoki, K. and Ueda, H.: Sieving Using Bucket Sort, *ASIACRYPT 2004*, LNCS 3329, pp.92–102, Springer (2004).
4) Bahr, F., Boehm, M., Franke, J. and Kleinjung, T.: RSA200, E-mail announcement (May 2005). http://www.loria.fr/~zimmerma/records/rsa200
5) Bajracharya, S., Misra, D., Gaj, K. and El-Ghazawi, T.: Reconfiguable Hardware Implementation of Mesh Routing in Number Field Sieve Factorization, *FPT 2004*, IEEE, pp.263–270 (2004).
6) Bernstein, D.: Circuits for Integer Factorization: A Proposal, preprint (2001).
7) Cavallar, S., Dodson, B., Lenstra, A., Lioen, W., Montgomery, P., Murphy B., te Riele, H., Aardal, K., Gilchrist, J., Guillerm, G., Leyland, P., Marchand, J., Morain, F., Muffett, A., Putnam, C. and Zimmerman, P.: Factorization of a 512-bit RSA Modulus, *EUROCRYPT 2000*, LNCS 1807, pp.1–18, Springer (2000).
8) Franke, J., Kleinjung, T., Paar, C., Pelzl, J., Priplata, C., Šimka, M. and Stahlke, C.: An Efficient Hardware Archtecture for Factoring Integer with the Elliptic Curve Method, *SHARCS 2005*, ECRYPT (2005).
9) Franke, J., Kleinjung, T., Paar, C., Pelzl, J., Priplata, C. and Stahlke, C.: SHARK: A Realizable Special Hardware Sieving Device for Factoring 1024-bit Integers, *SHARCS 2005*, ECRYPT (2005).
10) Franke, J., Kleinjung, T., Paar, C., Pelzl, J., Priplata, C. and Stahlke, C.: SHARK: A Realizable Special Hardware Sieving Device for Factoring 1024-bit Integers, *CHES 2005*, LNCS 3659, pp.119–130 (2005).
11) 富士通：素因数分解装置の調査・検討に関する報告書，暗号技術関連の調査報告（2003 年度），情報処理振興事業協会（IPA），通信・放送機構（TAO）(Feb. 2004).
12) 富士通，富士通研究所，情報通信機構：世界初，専用ハードウェアによる素因数分解実験に成功，プレスリリース (Sep. 2006).
13) FUJITSU, FUJITSU Lab., NICT: Factor-

ing c128 in $7^{352}+1$ by using special sieving hardware, E-mail announcement (Sep. 2006). http://www.loria.fr/~zimmerma/records/c128
14) Gaj, K., Kwon, S., Baier, P., Kohlbrenner, P., Le, H., Khaleeluddin, M. and Bachimanchi, B.: Implementing the Elliptic Curve Method of Factoring in Reconfigurable Hardware, *SHARCS 2006*, ECRYPT (2006).
15) Gaj, K., Kwon, S., Baier, P., Kohlbrenner, P., Le, H., Khaleeluddin, M. and Bachimanchi, B.: Implementing the Elliptic Curve Method of Factoring in Reconfigurable Hardware, *CHES 2006*, LNCS 4249, pp.119–133, Springer (2006).
16) Geiselmann, W., Januszewski, F., Köpfer, H., Pelzl, J. and Steinwandt, R.: A Simpler Sieving Device: Combining ECM and TWIRL, Cryptology ePrint Archive 2006/109, IACR (2006).
17) Geiselmann, W., Januszewski, F., Köpfer, H., Pelzl, J. and Steinwandt, R.: A Simpler Sieving Device: Combining ECM and TWIRL, *ICISC 2006*, LNCS 4296, pp.118–135, Springer (2006).
18) Geiselmann, W. and Steinwandt, R.: A Dedicated Sieving Hardware, *PKC 2003*, LNCS 2567, pp.254–266, Springer (2003).
19) Geiselmann, W. and Steinwandt, R.: Hardware to Solve Sparse Systems of Linear Equations over GF(2), *CHES 2003*, LNCS 2779, pp.51–61, Springer (2003).
20) Geiselmann, W. and Steinwandt, R.: Yet Another Sieveing Device, Cryptology ePrint Archive, 2003/ 202, IACR (2003).
21) Geiselmann, W. and Steinwandt, R.: Yet Another Sieveing Device, *CT-RSA 2004*, LNCS 2964, pp.278–291, Springer (2004).
22) Geiselmann, W. and Steinwandt, R.: Nonwafer-scaleSieving Hardware for the NFS: Another Attempt to Cope with 1024-bit, Cryptology ePrint Archive, 2006/403, IACR (2006).
23) Geiselmann, W. and Steinwandt, R.: Nonwafer-scaleSieving Hardware for the NFS: Another Attempt to Cope with 1024-bit, *EUROCRYPT 2007*, LNCS 4515, pp.466–481, Springer-Verlag (2007).
24) 廣田直之，國廣　昇，伊豆哲也，太田和夫：ルーティングを用いた素因数分解ハードウェアの 1024-bit 合成数分解に対する性能評価，*SCIS 2006*, 2E1-4 (2006).
25) Hirota, N., Izu, T., Kunihiro, N. and Ohta, K.: An Evaluation of the Sieving Device YASD for 1024-bit Integers, *SHARCS 2006*, ECRYPT (2006).
26) Hirota, N., Izu, T., Kunihiro, N. and Ohta, K.: An Evaluation of the Sieving Device YASD for 1024-bit Integers, *JWIS 2006* (2006).
27) 日立製作所：素因数分解専用集積回路等の実現性についての評価，暗号技術関連の調査報告（2003 年度），情報処理振興事業協会（IPA），通信・放送機構（TAO）(Feb. 2004).
28) McIvor, C., McLoone, M., McCanny, J., Daly, A. and Marnane, W.: Fast Montgomery Modular Multiplication and RSA Cryptographic Processor Architectures, *37th IEEE Computer Society Asilomar Conference on Signals, Systems and Computers*, pp.379–384 (2003).
29) 伊豆哲也，木田祐司：素因数分解の現状について，日本応用数理学会論文誌，Vol.13, No.2, pp.289–304 (2003).
30) Izu, T., Katou, K., Kogure, J., Nishimura, S. and Shimoyama, T.: An Implementation of a Sieving Algorithm in the Number Field Sieve Method on a Dynamic Reconfigurable Processor, *JWIS 2006* (2006).
31) Izu, T., Kogure, J. and Shimoyama, T.: CAIRN2: An FPGA Implementation of the Sieving Step in the Number Field Sieve Method, to appear in the *Proc. CHES 2007* (2007).
32) Kleinjung, T.: Cofactorisation Strategies for the Number Field Sieve and an Estimate for the Sieving Step for Factoring 1024-bit Integers, *SHARCS 2006*, ECRYPT (2006).
33) Kim, H. and Mangione-Smith, W.: Factoring Large Numbers with Programmable Hardware, *FPGA 2000*, pp.41–48, ACM (2000).
34) Lenstra, Jr., H.: Factoring Integers with Elliptic Curves, *Ann. Math.*, Vol.126, pp.649–673 (1987).
35) Lenstra, A. and Lenstra, Jr., H.: The Development of the Number Field Sieve, LNM 1554, Springer (1993).
36) Lenstra, A., Lenstra, Jr., H., Manasse, M. and Pollard, J.: The Number Field Sieve, *STOC 1990*, pp.564–572, ACM (1990).
37) Lenstra, A. and Shamir, A.: Analysis and Optimization of the TWINKLE Factoring Device, *EUROCRYPT 2000*, LNCS 1807, pp.35–52, Springer (2000).
38) Lenstra, A., Shamir, A., Tomlinson, J. and Tromer, E.: Analysis of Bernstein's Factorization Circuit, *ASIACRYPT 2002*, LNCS 2501, pp.1–26, Springer (2002).
39) Lenstra, A., Tromer, E., Shamir, A., Kortsmit, W., Dodson, B., Hughes, J. and Leyland, P.: Factoring Estimates for a 1024-bit RSA Modulus, *ASIACRYPT 2003*, LNCS 2894, pp.55–74, Springer (2003).
40) Pelzl, J., Šimka, M., Kleinjung, T., Franke, J., Priplata, C., Stahlke, C., Drutarovskỳ, M.,

Fischer, V. and Paar, C.: Area-time Efficient Hardware Archtecture for Factoring Integers with the Elliptic Curve Method, *IEE Proc. Information Security*, Vol.152, No.1, pp.67–78, IEE (2005).
41) Shamir, A.: Factoring Large Numbers with the TWINKLE Device, *CHES 1999*, LNCS 1717, pp.2–12, Springer-Verlag (1999).
42) Shamir, A. and Tromer, E.: Factoring Large Numbers with the TWIRL Device, *CRYPTO 2003*, LNCS 2729, pp.1–26, Springer (2003).
43) Shimoyama, T., Izu, T. and Kogure, J.: Implementation of a Sieving Algorithm on a Dynamic Reconfigurable Processor, *SHARCS 2005*, ECRYPT (2005).
44) 下山武司，伊豆哲也，小暮 淳：数体篩法による素因数分解専用ハードウェア装置の開発および実験，*SCIS 2007*, 3A2-3 (2007).
45) 下山武司，伊豆哲也，小暮 淳，西村 聡，加藤清光：数体篩法による素因数分解アルゴリズムのハードウェア DAPDNA2 への実装，*SCIS 2006*, 2E1-2 (2006).
46) Šimka, M., Pelzl, J., Kleinjung, T., Franke, J., Priplata, C., Stahlke, C., Drutarovskỳ, M., Fischer, V. and Paar, C.: Hardware Factorization Based on Elliptic Curve Method, *FCCM 2005*, pp.107–116, IEEE (2005).
47) Tenca, A. and Koç, C.: A Scalable Architecture for Montgomery Multiplication, *CHES 1999*, LNCS 1717, pp.94–108, Springer (1999).

(平成 18 年 11 月 27 日受付)
(平成 19 年 6 月 5 日採録)

伊豆 哲也（正会員）

1967 年生．1992 年東京大学理学部数学科卒業．1994 年立教大学大学院理学研究科数学専攻博士前期課程修了．1997 年立教大学大学院理学研究科数学専攻博士後期課程退学．博士（工学）．1997 年より富士通株式会社および株式会社富士通研究所に勤務，現在に至る．情報セキュリティ，暗号理論の研究に従事．2001 年 Waterloo 大学（カナダ）客員研究員．1999 年暗号と情報セキュリティシンポジウム（SCIS 1999）論文賞受賞．2002 年コンピュータセキュリティシンポジウム（CSS 2002）優秀論文賞受賞．2005 年電子情報通信学会基礎・境界ソサイエティ功労賞受賞．2007 年科学技術分野の文部科学大臣表彰若手科学者賞受賞．電子情報通信学会，IACR 各会員．

國廣 昇

1971 年生．1996 年東京大学大学院工学系研究科計数工学専攻修士課程修了．同年日本電信電話（株）入社．1996 年より 2002 年まで NTT コミュニケーション科学基礎研究所に勤務．2001 年東京大学より，博士（工学）授与．2002 年電気通信大学講師．2006 年同大学助教授，2007 年同大学准教授，現在に至る．博士（工学）．情報セキュリティ，暗号理論の研究に従事．著書に，『ほんとうに安全？現代の暗号』（岩波科学ライブラリー）．翻訳書に，『暗号理論』（岩波，1 冊でわかるシリーズ）．1997 年 SCIS'97 論文賞受賞．電子情報通信学会，IACR 各会員．

下山 武司

愛知県名古屋市出身．1989 年，1991 年に横浜市立大学にてそれぞれ学士ならびに修士取得．1999 年中央大学にて学位（工学）取得．1997 年 SCIS'97 論文賞受賞．1991 年から現在まで富士通株式会社および株式会社富士通研究所に勤務．ただし 1996 年から 1998 年まで通信放送機構に出向．主な研究テーマは暗号解析，素因数分解．2004 年 4 月および 2006 年 1 月特殊数体篩法による素因数分解世界記録達成．2006 年 5 月一般数体篩法による素因数分解世界記録達成．