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

2004－ＣＳＥＣ－26 (12)
2004／7／20

素因数分解ハードウェア TWIRL の実現可能性に関する検討報告(II)

伊豆 哲也† 伊藤 孝一† 小暮 淳† 小檜山 清之† 下山 武司†
武仲 正彦† 鳥居 直哉† 桝井 昇一† 向田 健二†

†(株) 富士通研究所/富士通株式会社

あらまし TWIRL とは Adi Shamir と Eran Tromer によって 2003 年 8 月に提案された素因数分解専用ハードウェアデザインの名称で，数体篩法における篩ステップを実現している．彼らの試算では，1024-bit 合成数を素因数分解する(のに必要な篩ステップを処理する) のに，1000 万ドル(約 10 億円) の費用と約 1 年の計算時間が必要であるとされている．筆者らは TWIRL の動作仕様の詳細調査と基本回路設計を行い，提案者の主張するハードウェアデザインの回路デザインの見直しと回路規模の見積を行った上で，その実現可能性について報告する．

キーワード 素因数分解ハードウェア, TWIRL, 評価, CRYPTREC

A Feasibility Analysis of the Factoring Device TWIRL (Part II)

Tetsuya IZU†, Kouichi ITOH†, Jun KOGURE†, Kiyoshi KOHIYAMA†, Takeshi SHIMOYAMA†
Masahiko TAKENAKA†, Naoya TORII†, Shoichi MASUI†, and Kenji MUKAIDA†

† FUJITSU LABORATORIES LTD./FUJITSU LIMITED

Abstract This document reports an overview of our evaluation results of the dedicated factoring device TWIRL by reviewing the circuit design assuming the current state-of-art technologies.

Keyword Dedicated hardware for integer factorization, TWIRL, evaluation, CRYPTREC

1. はじめに

TWIRL は，2003 年 8 月にイスラエルの暗号学者 Shamir と Tromer によって提案された素因数分解専用ハードウェア[ST03]である．このハードウェアは，現在最も効率的な素因数分解アルゴリズムとして知られている数体ふるい法[LL93]において最も処理量が大きいことが知られている，ふるい処理を行うことを目的とする．TWIRL は大規模な並列処理で行うことで，極めて効率的なふるい処理を実現することが特徴である．

彼らの試算に基づけば，1024-bit の場合における TWIRL を用いた素因数分解に必要なコストは，1000 万ドル(約 10 億円)と約 1 年の計算時間である．RSA 暗号など，現在の公開鍵暗号はその安全性の根拠は 1,024-bit 以上の素因数分解が困難であるという前提に基づいており，もし TWIRL が彼らの試算どおりのコストで製造・動作が可能であれば，素因数分解に基づいた公開鍵暗号の安全性が崩壊する可能性がある．しかしこの見積もりは，提案者自身も認めるとおり概算値でしかないため，実現可能性の可否を含めた再検討の必要性が指摘されていた．[ST03][RSA03]

本書は，素因数分解ハードウェア TWIRL に関する実現可能性に関する検討結果を報告する[1]．TWIRL は，有理数的ふるいを行う Rational TWIRL と，代数的ふるいを行う Algebraic TWIRL の 2 種類からなるが，本書においては，検討対象を Rational TWIRL の大部分を占める Largish/Smallish/Tiny Station とし，その他 Rational TWIRL の周辺回路および Algebraic TWIRL については対象外とした．以下本書は，TWIRL の中で最も規模が大きなブロックである Largish Station に関する検討報告を中心に行う．

なお，TWIRL の基本構造については，文献[ST03][F04_Ia]を，TWIRL の実装評価の詳細については文献[H04][F04_C]をそれぞれ参照されたい．

2. Largish Station

Largish Station は図 1に示す構造をもち，以下の 3 つのブロックから構成される．

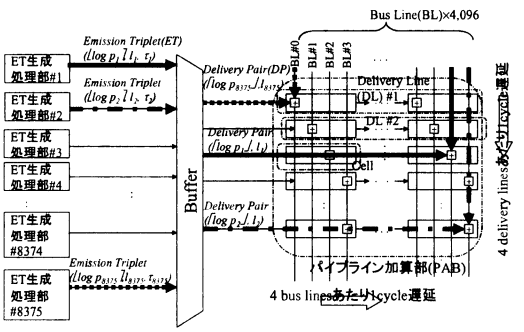
1. Emission Triplet 生成処理部(ETGB)
   Emission Triplet(ET) $(\lfloor log_2 p_i \rfloor, l_i, \tau_i)$を生成する．
2. Buffer

[1] 本研究は，通信・放送機構の委託研究「暗号の技術評価に関する研究開発」の一環として行ったものである．

ETGB から複数の ET を入力として受け取り，Delivery Pair (DP) $(\lfloor log_2 p_i \rfloor, l_i)$として出力する．データを出力する際には，パイプライン加算部における Delivery Line(DL)番号と出力タイミングの調整を，内部に格納されている複数の ET 間でスケジューリングすることで行う．

3. パイプライン加算部(PAB)
Buffer から DP$(\lfloor log_2 p_i \rfloor, l_i)$を入力として受け取ることで，パイプライン加算処理部の内部に格子状に配置されている cell において$\lfloor log_2 p_i \rfloor$の加算を行う．この処理は，(1)Delivery Pair の l_i が cell の番号と一致しないなら右の cell へ転送 (2)l_i が cell 番号と一致するなら，cell 内部のレジスタに$\lfloor log_2 p_i \rfloor$を加算，を各 cell が独立して繰り返すことで行う．(2)によって，$FS(a,b)$に$\lfloor log_2 p_i \rfloor$を加算する処理が行われる．

ETGB と Buffer と PAB は，ブロック間で連携してパイプライン的な動作を行うことで，極めて効率的なふるいを実現する．この連携を大雑把に言えば，ETGB から周期情報を ET として発行し，Buffer は複数の ET を一時的にストアしタイミング調整の上 Delivery Pair として出力し，PAB は Buffer から出力された Delivery Pair を元に素因数の対数値の加算処理を実行する．



・ET生成処理部(ETGB)#1~#8375: ET $(\lfloor log\, p_i \rfloor, l_i, \tau_i)$を出力 (1サイクル平均2,100 個)
・Buffer:l_i番目のBus Lineで時刻τ_iに$\lfloor log\, p_i \rfloor$が加算されるように、DP $(\lfloor log\, p_i \rfloor, l_i)$を発行
・パイプライン加算部:目的のCellに着くまでDPの内部転送を繰り返し、到着したら$\lfloor log\, p_i \rfloor$を加算

図 1 Largish Station のブロック図

これらのブロックが効率よく連携するためには，各ブロックが連携してパイプライン的な規則正しい動作を繰り返す必要がある．我々が TWIRL の文献[ST03]を調査した範囲では，ETGB と PAB については，提案者が主張する回路デザインを用いることで，パイプライン的な動作を実現可能であると考える．しかし，Buffer については，Shamir-Tromer は必要な機能については主張しているものの，機能の回路デザインは部分的にしか示していない．よって，彼らの主張する TWIRL のハード規模は（彼ら自身が主張する通り）目の粗いものである．また，Buffer において，Shamir-Tromer が回路デザインを示していない部分については，4096 通りの Bus Line(BL)番号に応じたソート処理であり，これは慎重にデザインを行う必要がある．例えば，その機能を典型的なバブルソート処理で実現した場合，5.6Ggates もの巨大な論理回路を必要とする．（完全なソートを実行する限り，回路規模が非常に巨大となることを避けるのは極めて難しいと考えられる．）我々はこのソート処理について検討を行い，小さな回路規模で実現できる回路構成を提案する．我々のデザインでは，非常に低い確率で発生する入力データの偏りにより，内部データオーバーフローが発生する代わりに，回路規模を 329Mgates まで抑えることができる．我々の回路デザインについて述べる前に，TWIRL 提案者による ETGB と PAB のデザインとそれぞれのブロックが実行するパイプライン的な動作について述べた上で，これらのブロックがパイプライン的に動作するために Buffer が果たすべき機能について述べる．

2.1. ETGB

ETGB のブロック図と動作概要を図 2に示す．ETGB は DRAM, SRAM, Processor と呼ばれる 3 つのモジュールから構成される．DRAM には，Progression Triplet(p_i, l_i, τ_i)と呼ばれるデータ(PT)が複数の p_i について格納される．（ただし，有効なデータは DRAM 全体のおよそ半分であり，残りは空．）紙面の都合上，本処理の詳細の言及は避け特徴のみを以下の 1.2.3 に示す．3.が Buffer の機能要件と関連している．なお，詳細については文献[F04_C]を参照されたい．

1. 集積度の高い DRAM 上に複数の p_i の情報を格納し，Processor が read/write を繰り返すことで，効率的な p_i の周期情報管理を実現．
2. DRAM(ランダムアクセスに弱いが回路規模が小さい)と SRAM(DRAM より回路規模が大きいがランダムアクセスに強い)の長所と短所を補っている．
3. 各 p_i に関する ET$(\lfloor log_2 p_i \rfloor, l_i, \tau_i)$は，「ほぼ」$p_i$ cycles ごとに出力される．つまり，理想的な出力タイミングより若干のずれがある．

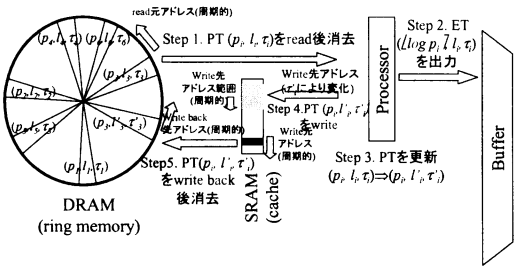


図 2 ETGB のブロック図

2.2. パイプライン加算部(PAB)

PAB の構造を図 3に示す．Cell と呼ばれるモジュールが格子状に配置されている．縦方向に並んだ一連の Cell を Bus Line と呼び，横方向に並んだ一連の Cell を Delivery Line と呼ぶ．Largish Station では，512 本の Bus Line と 525 本の Delivery Line を持つ PAB が 8 個並んでいる．

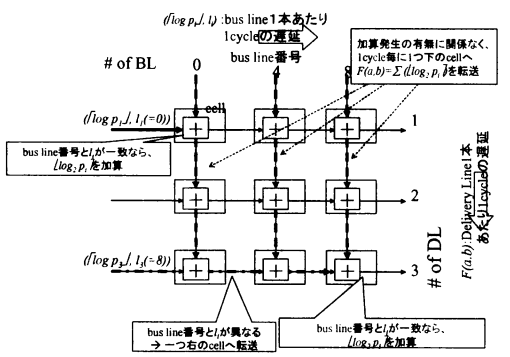


図 3 パイプライン加算部(PAB)のブロック図

PAB の各 Delivery Line の一番左の Cell からは，Delivery Pair($\lceil \log_2 p_i \rfloor$, l_i)が入力される．入力された Delivery Pair に対し，各 Cell は，l_i が Cell に対応する Bus Line の番号に一致するまで，一つ右の Cell にデータを転送することを繰り返す．l_i と Cell の Bus Line 番号が一致するならば，Cell 内部の加算器(10bit)にて，$\lceil \log_2 p_i \rfloor$を加算する．つまり，Delivery Pair が一つ右に移動するごとに 1 cycle の遅延が発生する．つまり，Delivery Pair が目的の Cell に到達するには，l_i の値に応じて決定されるデータ転送遅延を考慮して，Buffer から PAB に入力する必要がある．

同様に，Delivery Pair が加算される対象である $FS(a,b)$についても，各 Cell は 1cycle ごとに一つ下の Cell へ転送することを繰り返す．つまり，同一の Bus Line に対して$\lceil \log_2 p_i \rfloor$を加算する場合，例えば，ある時刻 τに Delivery Line#X に対して加算することが可能ならば，時刻 $\tau+u$ に Delivery Line#$X+u$ に加算することも可能である．この u の値を，複数の Delivery Pair の間で最適なスケジューリングを行うことで，加算器の稼働率を高め，ハードの使用効率を高めることができる．

以上により，パイプライン加算処理部の回路デザインを用いることで，各 Cell が独立して規則正しい処理を実行することで，パイプライン的な処理による効率の高い処理を実現することができる．ただし，このような高い処理効率を実現するためには，以下の 2 つの処理を Buffer 側で実施する必要がある．

1. l_i に関する調整機構．
2. Delivery Pair を出力する Delivery Line 番号と，Delivery Pair を出力するタイミングの調整．

2.3. Buffer

以上により，ETGB と PAB のパイプライン的な動作を実現するためには，Buffer に必要な機能は，以下の 3 つであり，TWIRL 論文[ST03]にもこれらの機能要件は主張されている．なお，TWIRL 論文は，(F1)(F2)(F3)の順に実行するように述べている．

(F1) 複数の Emission Triplet 間での，τ_i の値の調整．
(F2) 複数の Delivery Pair の間での，l_i の値ソート．
(F3) 複数の Delivery Pair の間での，出力する Delivery Line 番号とタイミングの調整．

これらのうち，Shamir-Tromer が回路デザインを示したのは，(F1)だけであり，(F2)(F3)についてはデザインを示していない．ただし，(F3)については，Smallish の Funnel の回路デザインを応用することで実現できると考えられる．この回路(DL 出力制御部)のデザインについては，2.6 節で述べる．

しかし，DL 出力制御部の処理が正しく実行されるのは，(F2)に示した，l_i に関するソート処理が行われている場合に限られる．よって，(F2)に示した l_i に関するソート処理の回路のデザインは，TWIRL 実現の検討において非常に重要な課題となる．しかし，TWIRL における l_i に関するソート処理のデザインは簡単ではない．なぜなら，Buffer は 4,096 通りの l_i の値を持つ Delivery Pair($\lceil \log_2 p_i \rfloor$, l_i)のデータのソート処理を，1cycle ごとにおよそ 2,100 個処理する必要があるから

である．この処理をハードウェアでは典型的に用いられるソート処理であるバブルソートを用いて実施する場合，5.6G gates 以上の非常に巨大な論理回路を必要とする．なぜなら，バブルソートによる処理は完了までに 4,096cycles 必要とするが，ソート対象の 2,100 個のデータは 1 cycle ごとに次々と入力される．よって，データ出力を 1 cycle ごとに行うためには，4,096cycles のバブルソート処理を 4,096 ステージのパイプライン処理を用いて実行する必要がある．結果，5.6Ggates 以上の非常に巨大な回路(90nm CMOS テクノロジーで 150mm×150mm のチップ面積を必要とする！)を必要とする．よって，ソート処理の回路デザインを見直し，回路規模を抑える構成にする必要がある．

我々は，この l_i に関するソート処理回路(l-sort 回路)について検討を行い，329M gates で処理できる回路デザインを提案する．次の 2.4 節では，提案する回路デザインについて説明する．

2.4. *l*-sort 回路

本節では，我々が検討を行った，l-sort 回路の機能を 329M gates で処理できる回路デザインを提案する．我々の回路デザインを説明するために，Buffer 全体のブロック図を図 4に示した[2]うえで，(F2)について説明する．なお，(F1)は[ST03]を，(F3)は 2.6 節をそれぞれ参照されたい．

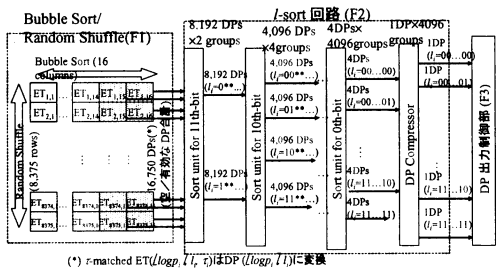


図 4 Buffer のブロック図

(F1)では，τ_i に関する選別処理を行う．これは，それぞれの ET に対し row 方向に Bubble Sort を，column 方向に Random Shuffle を実行し，右端の 2 columns のうち τ_i の値が特定のものを選別し(τ-matching)，(F2)に対して出力する．このとき，ET($\lfloor log_2 p_i \rfloor, l_i, \tau_i$)から DP($\lfloor log_2 p_i \rfloor, l_i$)への変換が行われる．(F1)から(F2)へ送られる 16,750 の ET は，空／有効なデータの両方を含んでいるおり，そのうち 2,100 個の ET が有効である．(F2)では，空／有効を含んだ 16,750 個のデータに対し，ソート処理が行われる．このソート処理は，l_i の k 番目のビット値に応じて，入力データを 2 つのグループに分類するユニットを用いる．我々はこのユニットを "Sort unit for k-th bit"と呼ぶ．Sort unit for 11-th bit により，16,750 個の ET が，l_i=0**...(もしくは空)である 8,192 個の ET のグループと，l_i=1**...(もしくは空)である 8,192 個の ET のグループの 2 グループに分類される．以下同様に，"Sort unit for k-th bit"を k=10,9,...,0 のそれぞれが，2W 個のデータを 2 つの W 個のグループに分類することを繰り返すことで，最終的に 4ET×4,096groups となるように分類する．4,096 通りに分類されたデータは，DP Compressor によって，各グループ(最大)4 つ DP の $\lfloor log_2 p_i \rfloor$ の値を加算することで，一つの DP にまとめる．

以上における我々の回路デザインにおいて基本的な処理となるのは，"Sort module for k-bit"の処理である．しかし，16,000 以上の ET を 2 つのグループに分類する処理を，一つのモジュールで処理するのは非常に巨大な回路規模を必要とする．つまり，16,000 個の ET を M 個ごとの section に分類し，各 section に含まれる M 個の ET を，l_i の k 番目のビット値に応じて 2 つの N 個のデータに分類することである．我々は，この機能を実現する回路を,"M-N×2 divider for k-th bit"と呼び，次のセクションで回路デザインについて説明する．なお，用いる M-N×2 divider for k-th bit の個数について述べると，k=11 の場合 8-4×2 divider を 1,865 個，10-4×2 divider を 183 個(k=11)であり，k=10,9,...0 の場合 8-4×2 divider を 2,048 個である．

2.5. M-N×2 divider for k-th bit

本節では，M-N×2 divider の回路デザインと動作について説明する．これは，(大部分を占める)M=8, N=4 について，k=0 の場合の動作例を示した図 5を用いて説明する．図 5に示した例では，8 個の入力データ DPin#1~DPin#8 のうち，k=0，すなわち l_i の LSB 値=0 であるデータが DPin#1(l_i=1,332)と DPin#8(l_i=540)に格納されており，l_i の LSB 値=1 であるデータが DPin#2(l_i=2013)に格納されている．(その他の DPin のデータは空である) これら 8 個のデータ入力に対し，LSB 値=0 であるものを DPout0#1~#4 に，LSB 値=1 であるものを DPout1#1~#4 にそれぞれ出力する．この処理を実現するためには，DPin#X が持つ DP のデータが，

[2] Bubble Sort/Random Shuffle(F1)については，Shamir-Tromer が row と column の数を明示していなかったが，我々は 8,375 rows, 16columns としてデザインした．

左端(つまり DPin#1)から数えて，何番目の l_i=0(もしくは l_i=1)であるデータであるかどうかをカウントし，カウントした値に応じて出力先の DPout0(もしくは DPout1)の番号選択することで実現できる(この選択処理は図 5の 1-4 DEMUX#0~1,1~4 により行われる)．この例では，DPin#1 は l_i=0 である 1 番目のデータ，DPin#8 は l_i=0 である 2 番目のデータ，DPin#2 は l_i=1 である 1 番目のデータである．このように，l_i のビット値に応じたデータ数のカウント処理は，Bit Slice Adder(BSA)により実現する．BSA は，s 個の 1bit 値を並列に加算し，t-bit 値として出力する加算器のことであり，$(s$-$t)$ 個の Full Adder を用いて実現できる．

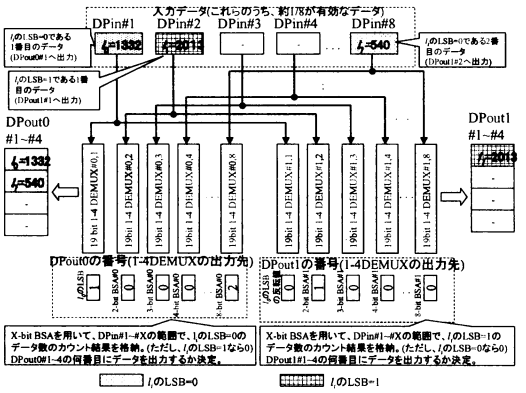


図 5 M-N×2 divider for k-th bit のデザイン (M=8,N=4,k=0)

なお，入力データに偏りがある場合，DPout0 もしくは DPout1 へ転送される対象となるデータの個数が 4 個を超え，このオーバーフローによりデータが損失される可能性がある．その確率は，$\Sigma_{k=5,\ldots,M}({}_MC_k \times q^k \times (1-q)^{M-k})$により与えられる．ただし，$q$ は，有効なデータが入力される確率 q'を用いて $q=q'/2$ で与えられる値である．k=11 の場合 q=0.0625 であり，M=8 ならば確率は 0.0000455，M=10 ならば確率は 0.000184 である．k=10,9,...,0 の場合 q=0.064，M=8 から，確率は 0.0000514 である．これらより k=11,10,...0 全体でデータを失う確率は 0.000737 と評価される．この確率は十分小さく，ふるいを行う上で問題ない範囲であると考えられる．

2.6. DL 出力制御部

ブロック図を図 6に示す．このブロックの機能は，PAB 内部 DP の転送遅延と，$v(a,b)$の転送遅延を調整することである．この調整を複数の DP 間でスケジュールすることで，PAB のほぼ全ての加算器の利用効率を高めた並列ふるい処理を実現する．このスケジュールを説明する．これは，Smallish Station における配送機構である Funnel のデザインを流用した．

入力データ 512 個の DP は，h 番目の DP が PAB に入力されてから目的の Bus Line に到達するまでに h サイクルかかるように，(l-sort 回路により)配列されている．本ブロックに入力された DP は，図 6に示すように，512×525 の平行四辺形の構造で保持される (Pipelined Network)．この内部では図 6の 1.2.に示す処理により DP 間のスケジュールが実現される．

1.は DP の転送遅延の調整を行う．2.は $v(a,b)$の転送遅延の調整を行う．さらに 2.により，空である DP を含む疎な DP ベクトルを入力しても，出力する DP ベクトルを密なベクトル列とすることができる．よって，PAB 内部の加算器の利用効率を高めることができる．

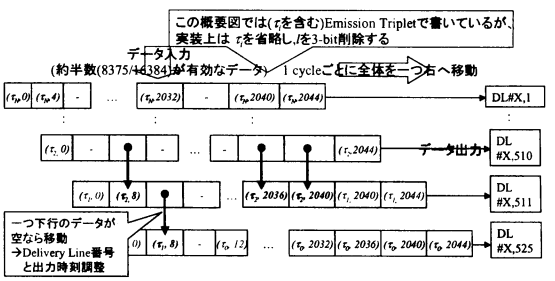


図 6 DL 出力制御部のブロック図

3. TWIRL の回路規模評価結果

Rational TWIRL の回路規模の評価結果を Table1 に示す．本評価は以下の前提で算出した我々の評価結果と，Shamir-Tromer[ST03]に示されていた 130nm CMOS テクノロジーによる回路規模見積を 90nm に換算した結果のそれぞれについて，必要な回路面積を比較することで行った．

1. 90nm CMOS テクノロジーを使用
2. 論理回路:4.0×10^{-6} mm^2/gate
3. SRAM:1.8×10^{-6} mm^2/gate
4. DRAM: 0.09×10^{-6} mm^2/bit

Table1. Rational TWIRL の回路規模見積

	Shamir-Tromer	本研究
Largish Station	5,814mm^2(76%)	14,935mm^2(63%)
Smallish Station	1,607mm^2(21%)	8,379mm^2(36%)
Tiny Station	228mm^2(3%)	218mm^2(1%)
合計	7,649mm^2	23,892mm^2

結果，Rational TWIRL の実装には 23,892mm^2 の面積が必要であり，Shamir-Tromer の見積である 7,649mm^2 と比較すると 3.12 倍必要であるとの結論となった．この差は，Largish Station の Buffer と，Smallish Station における Funnel の回路規模について，我々がレビューした結果が Shamir-Tromer の見積より大きいことによるものである．

4. 実現可能性の検討評価結果

3 章の回路規模見積結果より，Rational TWIRL を 1 個の LSI として製造するためには，90nm プロセスを用いた場合でも直径 100mm 以上のウェハーを必要とするが，そのように巨大なウェハーを製造することは現在の技術では不可能である．この問題を回避するために，現在の技術で製造可能な LSI を複数用いて，Rational TWIRL を実現するという前提のもとに，必要な LSI 数について検討を行った．なお，本書は検討評価結果の概要と結論のみを述べるものであり，その詳細については，文献[F04_C]を参照されたい．

4.1. 複数の LSI を用いた場合の実装検討

本検討にあたっては，各 LSI はハード規模と IO ピン数の両方について，以下の制約条件を守るものとして，90nm プロセスを用いた場合に必要となる LSI 数の見積を行った．

1. 回路規模に関する制約条件
   最大 1,600mm^2/LSI (400Mgates/LSI)
2. IO ピン数に関する制約条件
   最大 3,000pins/LSI(1,500pins を IO として使用可能．残り 1,500pins は Vcc/Gnd 用)

結論を述べると，回路規模に関する制約条件を用いると LSI を 15 個で実装可能であるが，IO ピン数に関する制約条件も考慮すると，3,362 個必要である．つまり，IO 数のボトルネックが原因で膨大な LSI が必要となる．また，これだけの個数の 40mm 角 LSI を 1 枚のボードに搭載するためには，58×58 個に並べる必要があり，LSI 間の配線領域に 10mm 必要と仮定するとおよそ 3m×3m の巨大なボードが必要となる．よって，2004 年 1 月現在で製造可能な複数の LSI を 1 枚のボードに搭載することで Rational TWIRL 実現を検討した場合，LSI 数が多すぎて製造不可能であり，この原因は IO 数のボトルネックによるものであると結論付けられる．

4.2. 複数のボードを用いた場合の実装検討

4.1 節の検討結果により，Rational TWIRL を実現するためには 3,362 個もの LSI が必要であり，LSI 数が多すぎて 1 枚のボードでは実現不可能であるとの結論を得た．よって，さらなる TWIRL 実現可能性の検討として，複数のボードを用いた場合の TWIRL の実現可能性について考察する．本検討における実現可能性検討方針は以下の通りである．

1. 1 枚のボードあたり 100 個の LSI を搭載可能する．
2. 分割にあたっては，ボード間の IO 数をなるべく少なくする方法を採用する．
3. 2.の結果，ボード間の IO 数が最大となる箇所により実現可能性を判断する．
4. ボード間の IO 数が多いほど IO の高速化が難しくなることを考慮すると，ボード間 IO が 400bit を超えるなら，TWIRL 提案者の主張する動作周波数(1GHz)を実現することは，現在の技術では不可能と判断する．

結論を述べると，パイプライン加算部におけるモジュール間の結合が密であることが原因により，ボード間 IO を 1,500bit 以下にするのは不可能であるため，上記の 4.に従い実現不可能との結論を得た．

5. TWIRL 実現に必要なブレークスルー

4 章の議論により，現在の半導体テクノロジーで実現可能な技術を用いた場合，TWIRL を実際に実現することは極めて困難との結論を得た．しかし将来の技術進歩によって実現できる可能性は残されている．4 章においては，TWIRL の実現にあたっては LSI 間およびボード間の IO 数がボトルネックとなっていた．よって TWIRL を実現するためには IO を高速かつ多ビット化する技術が必須であると考えられる．以下では IO を高速・多ビット化する技術の例と，それぞれの技術における問題点について簡単に説明する．

- シリアル-パラレル変換技術の高性能化
  IO ピン数を仮想的に増加させる技術としてシ

リアル-パラレル変換(シリ-パラ変換)が知られており，Rational TWIRL の場合，必要な LSI 数は LSI あたりの IO 数の 2 乗に反比例して小さくなることから，シリ-パラ変換を適用することで必要 LSI 数を減らすことができると考えられる．しかしその代償として，LSI 間 IO の周波数を上げる必要が生じる．例えば 10:1 のシリ-パラ変換を用いた場合，LSI 間は内部の 10 倍の周波数で動作させる必要があり，TWIRL 提案者の主張に従い LSI 内部を 1 GHz で動作させた場合，LSI 間 IO は 10GHz が必要となる．現在実用化されている 10GHz I/F の消費電力を考慮すると，TWIRL の LSI 数の問題を解決できるほど多くの IO 数について適用することは，当面は困難であろう．

- Active Substrate Multi-chip Module Package
  チップ間配線の結線を微細化することによって，プリント版配線を用いた場合より IO 数を増加させることが可能となる．このような技術として" Active Substrate Multi-chip Module Package" [Cha94] が知られている．しかし技術は論文レベルであり，実用レベルでの適用例は知られていない．本技術が現実的に使用可能であるか，さらには TWIRL に適用可能かであるかに関してはさらなる検討が必要である．

6. まとめ

素因数分解ハードウェア TWIRL の実現可能性に関する調査検討結果を報告した．本検討は Rational TWIRL に関して，Shamir-Tromer が提案した回路デザインについてレビューを行い，回路規模を算出したうえで，実装方法について検討することで行った．

回路デザインをレビューした結果，Largish Station における Buffer の一部の機能について Shamir-Tromer の回路デザインが提案されていなかったため，我々がデザイン案を検討した．

回路規模評価を 90nm CMOS テクノロジーの場合について行った結果，Rational TWIRL 全体で 23,892mm^2 もの面積が必要であり Shamir-Tromer 見積の 3.12 倍もの回路面積が必要であるとの結論を得た．

検討した回路デザインを，現在の半導体テクノロジーで生産可能な LSI(40mm×40mm 角，3000 IO pins)を使用することを前提に実装検討を行った結果，LSI あたりの IO の制約により 3,362 個の LSI が必要であり，1 枚のボード上には搭載困難であること，また複数のボードを用いた場合でもボード間 IO が多すぎて(1,500bit)Shamir-Tromer が主張する性能(1GHz)で動作させることは困難であるため，現在の半導体技術レベルでは TWIRL の実現は困難であるとの結論を得た．

よって，TWIRL 実現のボトルネックは IO 性能であり，TWIRL 実現のためのブレークスルーとしては，シリ-パラ変換の高性能化，Active Substrate Multi-chip Module Package[Cha94]を用いた LSI 間 IO の高性能化が挙げられる．

謝辞

本研究は，通信・放送機構の委託研究「暗号の技術評価に関する研究開発」の一環として行ったものである．

文　献

[Cha94] Chin-Chieh Chao, "Multiport Memory Design for an MCM Coprocessor", Technical Report No.ICL94-037, pp.95-113, Department of Electrical Engineering Stanford Univ., December 1994.

[CRY02] 情報処理振興事業協会(IPA), 通信・放送機構(TAO), "暗号技術評価報告書(2002 年度版)", March, 2002. Availabe at http://www.shiba.tao.go.jp/kenkyu/CRYPTREC/PDF/c02_report.pdf

[F+03] J. Franke and others, "RSA-576", Email announcement, December 2003. Available at http://www.crypto-world.com/announcements/rsa576.txt

[F04_C] (株) 富士通研究所/富士通株式会社, "素因数分解装置の調査・検討に関する報告書" 2004 年 2 月. Available at www.ipa.go.jp/security/enc/CRYPTREC/fy15/documents/rep_ID0208.pdf

[F04_Ia] 伊豆, 伊藤, 小暮, 小檜山, 下山, 武仲, 鳥居, 桝井, 向田, "素因数分解ハードウェア TWIRL の実現可能性に関する検討報告(I)" 電子情報通信学会技術報告書, ISEC 2004 (2004-07), July 2004.

[H04] (株) 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" 2004 年 2 月. Available at www.ipa.go.jp/security/enc/CRYPTREC/fy15/documents/rep_ID0209.pdf

[IK03] 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.

[LL93] Arjen K. Lenstra and H.W. Lenstra (eds.), "The development of the number field sieve", Vol. 1554 in Lecture Notes in Mathematics (LNM), Springer-Verlag, 1993.

[LSTT02] Arjen K. Lenstra, Adi Shamir, Jim Tomlinson

and Eran Tromer, " Analysis of Bernstein's circuit", ASIACRYPT 2002, LNCS 2501, pp.1-26, Springer-Verlag, 2002.

[LTS+03] Arjen K. Lenstra, Eran Tromer, Adi Shamir, Wil Kortsmit, Bruce Dodson, James Hughes and Paul Leyland, "Factoring estimates for a 1024-bit RSA modulus", ASIACRYPT 2003, LNCS 2894, pp.55-74, Springer-Verlag, 2003.

[RSA03] RSA Security, "TWIRL and RSA Key Size", May 2003. Available at http://www.rsasecurity.com/rsalabs/technotes/twirl.html

[ST03] Adi Shamir and Eran Tromer, "Factoring large numbers with the TWIRL device", CRYPTO 2003, LNCS 2729, pp.1-26, Springer-Verlag, 2003.