情報処理学会論文誌 Vol. 52 No. 12 3755–3765 (Dec. 2011)

推薦論文

ハッシュ関数 Luffa のハードウェア実装

片下敏宏[†1] 佐藤証[†1] 菅原健[†2]
本間尚文[†2] 青木孝文[†2]

本論文では，次世代ハッシュ関数 SHA-3 の候補として提案されたスポンジ関数型のアルゴリズム Luffa に対し，複数のハードウェア・アーキテクチャを提案し，90 nm CMOS スタンダードセル・ライブラリによる ASIC 実装および Xilinx Virtex-5 と Spartan-6 による FPGA 実装性能評価を行った．その結果 ASIC では，回路規模 14.7 K〜62.8 Kgates においてスループット 3.6 G〜35.1 Gbps となり，小型からきわめて高速な実装まで実現可能なことが分かった．また FPGA 実装でも同様に，Virtex-5 では 750〜1,548 Slices において 1.3 G〜7.0 Gbps，Spartan-6 では 592〜1,535 Slices において 1.3 G〜5.5 Gbps と，同様の実装性能が示された．さらに，同じスポンジ関数型であり SHA-3 候補の Keccak アルゴリズムと同条件において比較したところ，Luffa はスループットにおいて同等の性能を持ちつつ，小型実装においてはおよそ半分の回路規模となり，回路構成の柔軟性が高いことが分かった．このほか，データバス構成とデータ処理の独立性が演算回路共有の効果に影響することが分かり，ハッシュ関数の設計においてハードウェア実装ではデータ処理の並列性が重要であることが明らかとなった．

Hardware Implementations of Hash Function Luffa

Toshihiro Katashita,[†1] Akashi Satoh,[†1]
Takeshi Sugawara,[†2] Naofumi Honma[†2]
and Tatafumi Aoki[†2]

This paper presents hardware architectures of the hash algorithm Luffa proposed for the next generation hash standard SHA-3. The architectures were evaluated by using a 90 nm CMOS standard cell library and Xilinx Virtex-5 and Spartan-6 FPGA devices. The ASIC implementations achieved a variety of circuits, from compact to very high-speed; throughputs of 3.6 G–35.1 Gbps with hardware resources of 14.7 K–62.8 Kgates. The FPGA implementations also showed high performances; throughputs of 1.3 G–7.0 Gbps with hardware sizes of 750–1,548 Slices for Virtex-5, and throughputs of 1.3 G–5.5 Gbps with hardware sizes of 592–1,535 Slices for Spartan-6. In comparison with other SHA-3 candidate Keccak that belongs to a category of a sponge function as same as Luffa, Luffa showed advantages in flexibility from high-speed (comparable to Keccak) to compact (half size of Keccak) hardware implementations. The results also show that data bus structure and parallelism of processing effect in design flexible.

1. はじめに

ハッシュ関数とは，任意の長さのメッセージから偽造が計算量的に困難な固定長のハッシュ値を生成するアルゴリズムであり，データの改ざん防止やエラー検出などに用いられている．広く利用されているハッシュ関数 SHA-1 や MD5 に対する強力な攻撃法[1),2)] が提案されたことから，ハッシュ関数の研究が活発化し，米国国立標準技術研究所（NIST: National Institute of Standards and Technology）では 2012 年の標準化を目指して次世代ハッシュ関数 SHA-3[3),4)] のアルゴリズムの公募を開始した．一次評価において 2008 年 11 月に 64 件の提案から 51 件が通過し，さらに二次評価で 2009 年 7 月に 14 件に絞られ，三次評価を経て 2010 年 12 月に，BLAKE[5)]，Grøstl[6)]，JH[7)]，Keccak[8)–11)]，そして Skein[12)] が最終候補のアルゴリズムとして選定されている．

二次評価を通過した 14 候補の中には，日立製作所とベルギー Leuven Katholieke 大学による共同開発の Luffa[13)] が日本からの提案として唯一存在していた．Luffa は Keccak と同じ「スポンジ型」と呼ばれる構成を採用しており，これは攪拌と中間的な圧縮処理により一時的な値を生成し，メッセージブロックの末尾が入力された際に圧縮処理を行ってハッシュ値を算出するものである．メッセージブロックごとに攪拌と圧縮処理を繰り返す従来のハッシュ関数に比べ，データバス構成が簡素となるスポンジ型は高速なハードウェア実装に適しており，Luffa と Keccak は他の SHA-3 候補に対して数倍高いスループットを実現できる[14)–23)]．Keccak は，ブロック暗号の標準として最も広く利用されている AES（Advanced Encryption Standard）[24)] の開発者の 1 人である Daemen が設計に加わっており，多彩な

†1 産業技術総合研究所
National Institute of Advanced Industrial Science and Technology
†2 東北大学大学院情報科学研究科
Graduate School of Information Sciences, Touhoku University
本論文の内容は 2010 年 7 月の DICOMO2010 シンポジウムにて報告され，CSEC 研究会主査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．

安全性評価が実施された点が評価され最終候補として選ばれたが，同じスポンジ型構造を持ち Keccak の対抗である Luffa は高い実装性能を持ちながらも最終候補に残ることはできなかった．しかしながら Luffa は Keccak が苦手とする小型回路実装にも適した構造を有し，高速用途から組み込みアプリケーションまで柔軟な実装を可能とするアルゴリズム構造の提示は，ハッシュ関数を含む暗号アルゴリズムの今後の設計に実装の柔軟性という指針を提起したといえる．

本論文では，高スループットから少回路規模まで柔軟な回路実装を可能とする Luffa の特徴を活かした複数のハードウェア・アーキテクチャを提案する．そして，STMicroelectronics 社 90 nm CMOS スタンダードセル・ライブラリ[25] および，Xilinx 社の FPGA Virtex-5 と Spartan-6 [26),27)] を対象に，それらアーキテクチャに基づく回路を論理合成し，スループット，回路規模，回路効率の評価を行う．文献 14)–23) における SHA-3 候補の回路性能比較では各アルゴリズムが仕様書どおりに単純に実装されているのに対し，本論文の Luffa 回路には様々な最適化を適用している．衝突攻撃に対するセキュリティ強度が Luffa と同等となるハッシュ長の Keccak の回路を，同じ論理合成条件のもとで比較し，同じスポンジ構造でを持ちながらも性能に大きな差が出ることを明らかにする．そして，それらの差が生じる要因をデータ処理の独立性に依存する演算器の共有化という視点で考察し，小型回路から高速回路まで柔軟な実装に適したアルゴリズムの構成について論じる．

以下，2 章では Luffa のアルゴリズムについて概説した後，4 つのハードウェア・アーキテクチャを提案する．次に 3 章では比較対象とした Keccak のアルゴリズムと，アルゴリズム開発者により提示されたハードウェア・アーキテクチャを示す．4 章では，Luffa と keccak の回路性能をスタンダードセル・ライブラリを用いた ASIC（Application Specific Integrated Circuit）実装および FPGA 実装によって比較し，アルゴリズム構造に起因する特徴について考察する．

2. Luffa のアルゴリズムとハードウェア・アーキテクチャ

2.1 アルゴリズム

図 1 に，256 bit のハッシュ値を生成する Luffa アルゴリズム構造を示す．Luffa は SHA-3 の仕様に沿って 224 bit，384 bit，512 bit のハッシュ値にも対応しているが，本論文では 256 bit のハッシュ値の Luffa を回路実装の対象とする．

Luffa は 256 bit の中間値を保持する 3 つデータパスを有し，256 bit の 3 つの初期値 V_0〜V_2 から処理を開始し，Message Injection 関数 MI で 256 bit のメッセージ $M(i)$ を印加した後に，256 bit の非線形転置関数 Q_0〜Q_2 でデータの撹拌を繰り返し，最後に 3 つの中間値を排他的論理和（XOR）してハッシュ値 Z_0 を生成する．

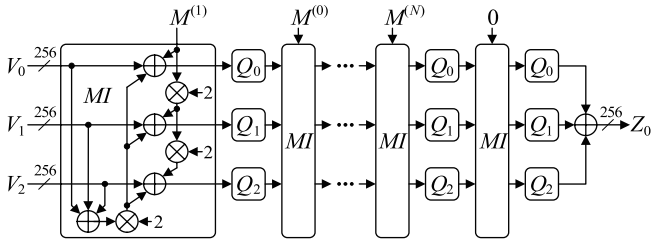

図 1　ハッシュ値 256 bit の Luffa のアルゴリズム
Fig. 1　The block diagram of Luffa-256.

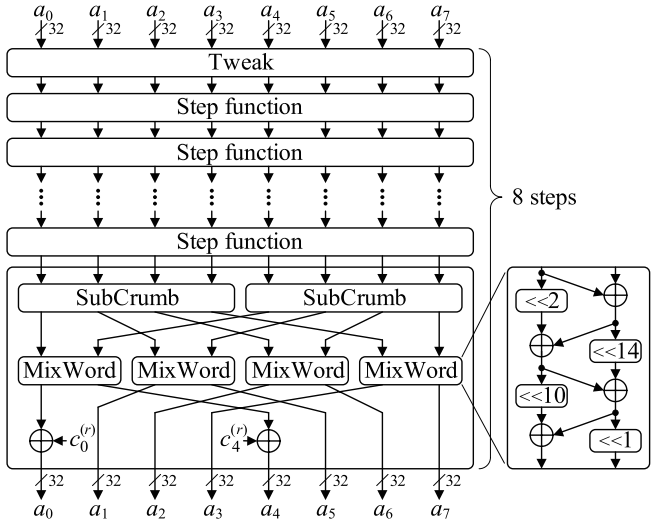

図 2　非線形転置関数 Q
Fig. 2　The block diagram of the Q function.

図 2 に各転置関数 Q 内の処理を示す．256 bit のデータは 32 bit バス a_0〜a_7 に分割された後，Tweak 処理によりビットローテーションが行われ，Step function が 8 回適用される．Step function は，32 個の 4 bit S-Box を有する SubCrumb 関数，2 つの 32 bit デー

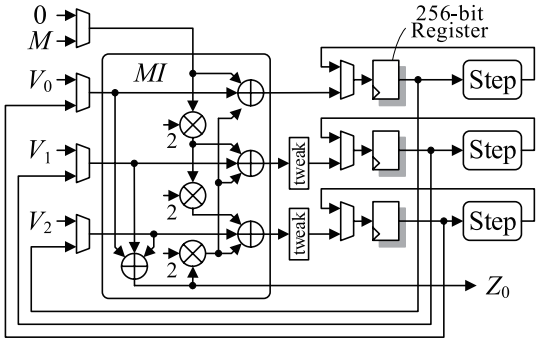

図 3　Luffa の標準型ハードウェア・アーキテクチャ
Fig. 3　The block diagram of the straightforward architecture.

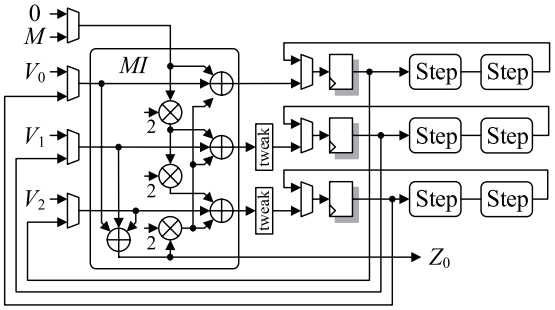

図 4　Luffa の高速型ハードウェア・アーキテクチャ
Fig. 4　The block diagram of the throughput-optimized architecture.

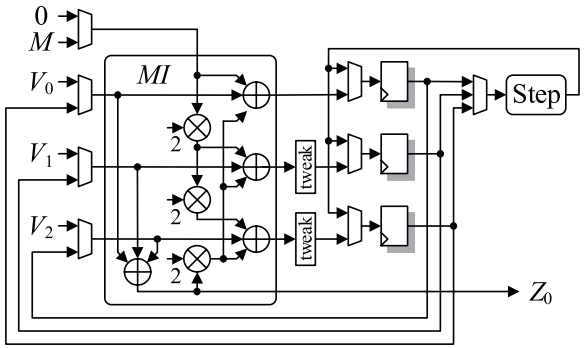

図 5　Luffa の共有型ハードウェア・アーキテクチャ
Fig. 5　The block diagram of the shared architecture.

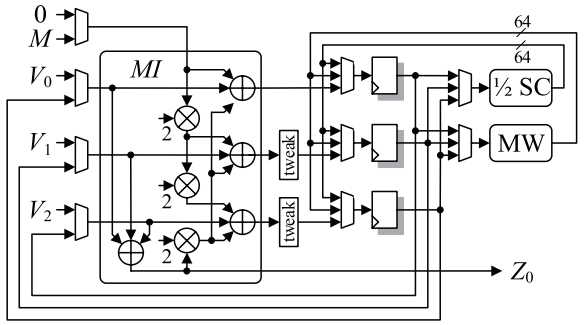

図 6　Luffa の小型ハードウェア・アーキテクチャ
Fig. 6　The block diagram of the size-optimized architecture.

タをローテーションし XOR 演算で撹拌する MixWord，そしてステップ r ごとに定められている 32 bit 定数 $C_0(r)$ および $C_4(r)$ との XOR 演算から構成されている．4 bit S-Box は同一の変換でありランダムな入出力テーブルとして定義されているほか，32 bit プロセッサによる処理を考慮したビットスライス実装も仕様書に示されている[13]．

2.2　ハードウェア・アーキテクチャ

Luffa の関数 Q や Step function は入出力ビット数の少ない関数を複数個配置した構成となっているため，それらを並列化あるいは共有化した回路構成をとることで速度と回路規模のトレードオフが選択できる．以下では，**図 3**，**図 4**，**図 5**，**図 6** に示す高速から小型まで 4 種類のハードウェア・アーキテクチャについて説明する．なお，図においてビット数の記述のないバス幅はすべて 256 bit となっている．

図 3 の標準型[14] は，図 1 のアルゴリズムを繰返し処理で実行する回路である．3 つの 256 bit バスそれぞれに Step function ブロックを備え，関数 Q を 3 並列で処理する構成であり，256 bit のメッセージが入力されるごとに Step function ブロックで演算が 8 回繰り返される．文献 14)–16) の回路と同様のアーキテクチャであるが，ハッシュ値 Z_0 の生成と関数 MI において XOR ゲートを共有することで 256 個の 3 入力 XOR ゲートが削減される．なお，XOR ゲートの共有は，図 4，5，6 中のすべてのアーキテクチャで採用している．

標準型では関数 MI に 1 サイクル，関数 Q に 8 サイクル要することから，256 bit のメッセージブロックの処理に $1+8=9$ サイクルを要する．

図 4 に示す高速型アーキテクチャは，データバスそれぞれに Step function ブロックを 2 個直列に備えることで，関数 Q の処理サイクルを標準型の半分の 4 サイクルとしている．Step function の追加により回路規模やクリティカルパスの遅延は増加するが，直列した Step function ブロックを論理圧縮することで，規模や遅延の増加を 2 倍以下に抑えることができる．また，中間値を保持するレジスタのセットアップ・ホールドのマージンは倍加しないため，遅延はさらに削減される．このアーキテクチャでは，1 メッセージブロックを $1+4=5$ サイクルで処理することができる．

図 5 は共有型アーキテクチャ[15] であり，1 つの Step function ブロックを 3 つのバスで共有して回路規模の削減を図っている．なお，3 つのバスでは関数 MI ブロック内部の 2 倍算器と XOR を共有することも可能であるが，これには中間値保持用のレジスタの増大だけでなく，サイクル数の増加も生じるため利点はない．このほか 3 つの関数 Q の演算が独立していることから，Step function ブロックを 3 分割してパイプライン化して動作周波数の向上を図る方式も考えられる．しかし，Step function のクリティカルパス遅延が小さいため効果は限定的である一方，パイプラインレジスタによるオーバヘッドが大きく利点はほとんどない．本アーキテクチャは標準型では 3 並列であった関数 Q を逐次処理するため，1 メッセージブロックの処理に $1+8\times3=25$ サイクルを要する．

最後に，図 6 に小型アーキテクチャを示す．ここでは，Step function 内の SubCrumb と MixWord を共有化して回路の削減を図っている．Step function 内の SubCrumb は 32 個の 4 bit S-box から構成されるため，独立な S-box 単位で演算を分割することが可能である．そこで，処理データ幅を MixWord に統一して 64 bit とし，SubCrumb は標準の半分である 16 個の S-box を持つ 64 bit のブロックを実装して共有化することとした．64 bit の MixWord（図 6 では MW と表記）と半分の SubCrumb（図 6 では 1/2 SC と表記）のブロックを 256 bit バスで共有するため，Step function の演算回路規模は 1/4 に削減される．なお，64 bit ごとに処理するための 4:1 マルチプレクサと 1:4 デマルチプレクサ，および制御回路が付加されるが，簡略のため図 6 では省略した．

Step function では SubCrumb と MixWord の間にローテーション処理が入るため，SubCrumb 処理がすべて完了するまで MixWord を実行できず逐次的な処理となる．したがって，単純に 64 bit のブロックを共有化すると，1 回の Step function の処理は SubCrumb の 4 サイクルと MixWord の 4 サイクルの計 8 サイクルとなり，1 メッセージブロックでは

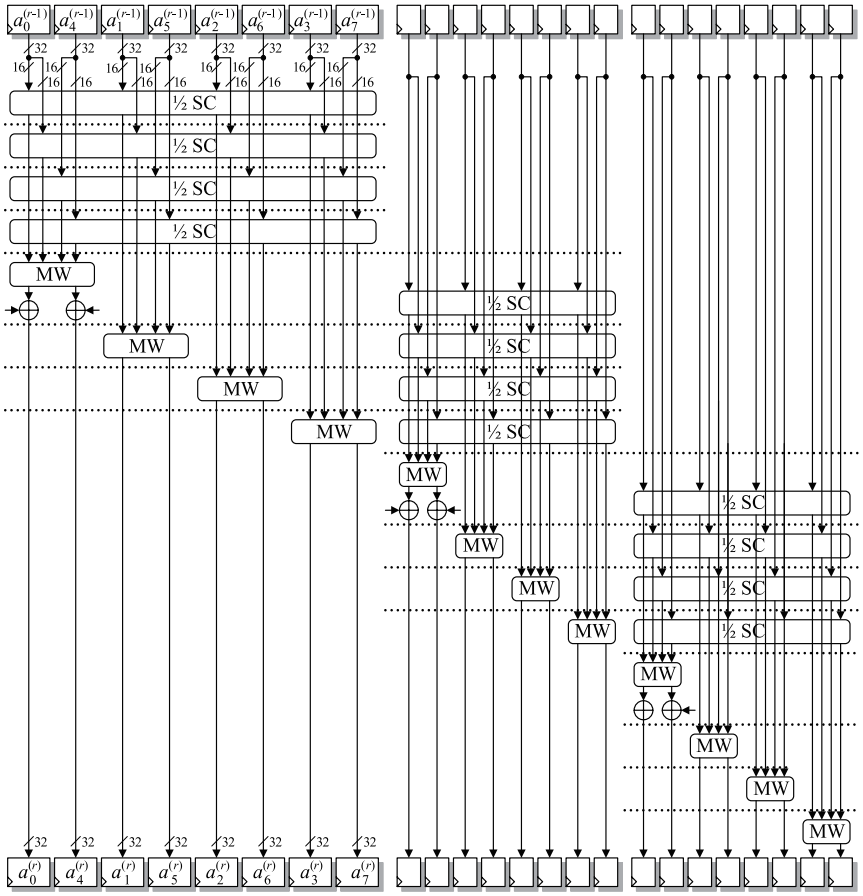

図 7　小型アーキテクチャにおける Step function のデータスケジューリング
Fig. 7　The procedure of Step function in the size-optimized architecture.

$1+(4+4)\times8\times3=193$ サイクルも要してしまう．そこで，3 つの関数 Q で 256 bit データが独立に演算できることを利用し，SubCrumb と MixWord において異なるデータバスを割り当てて並列処理することでサイクル数の短縮を図ることとした．**図 7** に並列処理のデータスケジューリングを示す．破線は各クロックサイクルを意味している．8 つの 32 bit ワード入力 a_0〜a_7 は，MixWord ブロック（MW）の入力に位置合わせされている．スケ

ジューリングにあたり，SubCrumb（1/2 SC）への入力は 32 bit ワード a_0〜a_7 を上位と下位の 16 bit に分割し，4 つずつ選択して MixWord の処理順に対応させている．図 7 より，3 つの 256 bit データバスに対して Step function を 1 回ずつ適用するには $4+4\times3=16$ サイクルを要するため，1 メッセージブロックでは $1+16\times8=129$ サイクルとなることが分かる．

3. Keccak アルゴリズムとハードウェア・アーキテクチャ

3.1 アルゴリズム

Keccak のデフォルトパラメータは，演算の中間値長 $b=1{,}600$，メッセージ長 $r=1{,}024$，安全性パラメータ長 $c=576$ で[10),11)]，このときハッシュ長 n は $b=r+c$，$c=2n$ より 288 bit となっている．SHA-3 向けのハッシュ長 256 bit に対する Keccak のメッセージ長は $r=b-2n=1{,}600-2\times256=1{,}088$ bit と 2 の累乗にならない．このため，Keccak のデフォルトパラメータは上記のように，メッセージ長を 2 の累乗の 1,024 bit とし，提案者の設計によるハードウェアコードもこれに準拠している[8)]．ハッシュ関数の衝突攻撃に対する計算量的な強度は，ハッシュ値のビット長で評価される．このためデフォルトのハッシュ長 288 bit の Keccak は 256 bit の Luffa より高い強度を持つが，本論文では Keccak の提案者によるハードウェアコードを比較対象とするため，両者をほぼ同等の強度と見なして回路性能の比較を行う．また，ハッシュ長の差が回路規模に及ぼす影響もあるが，以下でアルゴリズム構造の違いによる性能差を論じるうえで有意なものではない．**図 8** に示すように，Keccak はメッセージの印加と撹拌処理を繰り返す Luffa と同様の構造を有するが，メッセージの印加が XOR 演算のみと簡素な半面，撹拌処理 R が複雑である．処理 R は 5 種類の撹拌 (θ，ρ，π，χ，ι) を 24 回適用する構造で，1,600 bit の中間値を 1 ブロック 64 bit からなる 5×5 ブロックの 2 次元配列として扱う．

撹拌 θ，ρ，π では配列の回転操作が加わる[8)] ため，25 のブロックが相互に演算に影響を及ぼし，1,600 bit のデータパスを分割して独立に演算する回路構成をとることを妨げている．

3.2 ハードウェア・アーキテクチャ

図 9 に Keccak の開発者の設計による，標準的なハードウェア・アーキテクチャを示す．関数 R 内の繰返し処理は共有化されており，Luffa の標準型と同様の構造となっている．このアーキテクチャは，1,024 bit のメッセージブロックの処理に 24 サイクルを要する．しかし，関数 R 内の撹拌 θ，ρ，π でデータ配列全体に対する回転操作が加わるため，25 のブロックが相互に演算に影響を及ぼし，1,600 bit のデータパスを分割して独立に演算する回路構成をとることを妨げている．したがって，Luffa のようにデータブロックを分割・スケジューリングして回路を共有することができない．開発者は小型実装も行っている[10)] が，外部プロセッサが別途必要であり，1 ブロックごとに処理を行うためサイクル数は 5,160 ときわめて多い．これは命令拡張されたプロセッサによるソフトウェア実装と見ることもできるため，今回の比較においては対象外とした．なお，今回は比較実装を行っていないが，複数用意した R 関数ブロックを直列につなげ，サイクル数を削減する Luffa と同様の高速化手法が適用可能である[10)]．

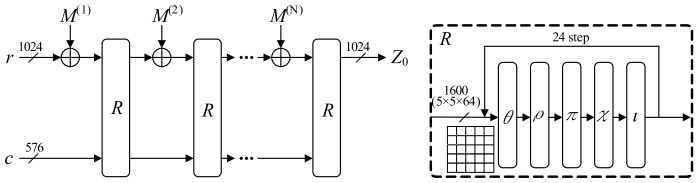

図 8　Keccak-1600（メッセージ長 1,024 bit）のアルゴリズム
Fig. 8　The block diagram of Keccak-1600.

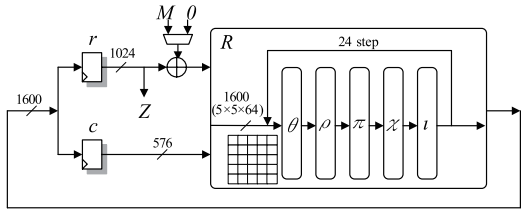

図 9　Keccak のハードウェア・アーキテクチャ
Fig. 9　The block diagram of the straightforward architecture of Keccak.

4. ハードウェア実装評価

2 章に示した Luffa の 4 つのアーキテクチャを Verilog-HDL 言語で設計し，ASIC 実装として STMicroelectronics 社の 90 nm CMOS スタンダードセル・ライブラリ[25)] を，FPGA 実装として Xilinx 社の Virtex-5[26)] および Spartan-6[27)] を用いて，スループット，回路規模，回路効率（単位回路あたりのスループット）の性能を評価した．なおこれらのソースコードは，文献 23) に示した Web サイトで公開を行っている．SubCrumb の S-box は 4 bit アドレスのテーブルとして表現した Table 型と，ビットごとに論理式で表現した Bit slice 型の 2

種類を用いた．また，Keccak はアルゴリズム提案者らが Web 上に公開[10] している 1,024 bit 標準型の VHDL ソースと，現在広く利用されてい標準ハッシュ関数の SHA-256[28),29)] も同じ合成条件で回路実装した．なお，これまで多数のハッシュ関数が採用している構造を持つ SHA-256 との比較は，新しいスポンジ型のハッシュ関数の回路実装面での特徴を参考情報として示すものである．

4.1 ASIC 実装

表 1 および **図 10** に，90 nm ライブラリを用いて Synopsys 社の Design Compiler version A-2007.12-SP3 により最悪条件下（1.08 V 125°C）で論理合成した Luffa，Keccak，そして SHA-256 の実装結果を示す．Luffa では 2 種類の S-box 実装の結果を示しているが，Keccak は S-box に相当する処理はなく論理演算で構成されているため 1 種類の実装となっている．使用ライブラリや論理合成の条件が異なることから直接の比較はできないが，表 1 には文献 14)，15)，19)，21) の Luffa 実装の結果と，文献 9)，14)，15)，19) の Keccak 実装の結果も参考情報として掲載している．表中の STMicro は ST Microelectronics 社のライブラリ，UMC は United Microelectronics Corporation のライブラリである．なお，演算コア部分だけの実装やマルチメッセージをパイプライン処理するものは，比較対象に含めていない．実装性能評価では，複数のタイミング制約によって合成した結果から，各アーキテクチャで動作速度が最大のもの，回路規模が最小のものに加え，回路効率が最大となった実装を記している．図 10 の横軸はゲート数（2 入力 NAND 換算），縦軸はスループットであり，グラフの左上にあるものほど回路効率が高いことを意味している．

表 1 において，高速型アーキテクチャで Table 構成の S-box を用いた実装が最大のスループット 35,069 Mbps となった．高速型では直列に並べた関数 Q の論理圧縮の効果により，最大動周波数は 684.9 MHz と標準型の 1,087 MHz に対して 28%の低下で抑えられている．しかしながら，関数 MI があるためにサイクル数は 5（$= 1 + 8/2$）となり，標準型の 9（$= 1 + 8$）サイクルの半分にはならず，スループットは標準実装の 31,258 Mbps に対して 12%の増加にとどまっている．また回路規模は 62,838 gates で，標準型の 39,513 gates に対して 59%の増加となった．なお，本評価では Luffa の高速型のスループットが最大であったが，Keccak も同様にラウンド処理を直列にすることが可能なため，高速化においては Keccak のほうが Luffa よりも優位であると考えられる．

回路規模最小は，Bit slice の S-Box を用いた共有型の 14,710 gates で，このときのスループットは 3,641 Mbps となった．共有型に対して，Step function の演算回路を 1/4 とした小型実装は，15,381 gates と共有型よりも大きくなり，スループットは 711.3 Mbps に減少

表 1　Luffa，Keccak，SHA-256 の回路実装性能（ASIC）
Table 1　Synthesis results of Luffa, Keccak and SHA-256 on ASIC.

アルゴリズム	ブロックサイズ (bits)	文献	実装方式 S-box	サイクル	回路規模 (gates)	動作周波数 (MHz)	スループット (Mbps)	回路効率 (kbps /gates)	プロセス
Luffa	256	本論文	高速	5	60,856	625.0	32,000.0	525.8	90 nm
			Bit		44,290	552.5	28,287.3	638.7	STMicro
			Slice		31,201	357.1	18,285.7	586.1	
			高速	5	62,838	684.9	35,068.5	558.1	
			Table		38,274	549.5	28,131.9	735.0	
					29,336	350.9	17,964.9	612.4	
			標準	9	39,394	1,000.0	28,444.4	722.0	
			Bit		19,736	552.5	15,715.2	796.3	
			Slice		18,907	357.1	10,158.7	537.3	
			標準	9	39,513	1,087.0	31,257.6	791.1	
			Table		19,604	549.5	15,628.8	797.2	
					18,933	355.9	10,122.6	534.7	
			共有	25	25,558	757.6	7,757.6	303.5	
			Bit		17,477	546.6	5,595.6	320.2	
			Slice		14,710	355.9	3,641.1	247.7	
			共有	25	26,373	862.5	8,462.8	320.9	
			Table		16,467	555.6	5,688.9	345.5	
					14,817	355.9	3,644.1	245.9	
			小型	129	24,285	813.0	1,613.4	66.4	
			Bit		16,801	552.5	1,096.4	65.3	
			Slice		15,381	358.4	711.3	46.2	
			小型	129	22,500	813.0	1,613.4	71.7	
			Table		16,633	555.6	1,102.5	66.3	
					15,383	358.4	811.3	46.2	
		14)	標準	9	44,972	483.1	13,741.2	305.6	180 nm
			Table	9	44,972	483.1	13,741.2	305.6	UMC
		15)	標準	8	55,000	727.0	23,256.0	422.8	90 nm
			共有	24	22,000	118.6	1,265.0	57.5	UMC
		19)	標準	9	26,551	245.0	6,968.9	262.5	130 nm
					37,942	490.0	13,937.8	367.3	UMC
		21)	標準	9	30,834	1,124.0	31,960.0	1,036.5	130 nm
			共有	26	18,260	250.0	2,461.5	134.8	UMC
Keccak	1,024	本論文	標準	24	50,675	781.3	33,333.3	657.8	90 nm
					33,664	540.5	23,063.1	685.1	STMicro
					29,548	354.6	15,130.0	512.0	
		9)	標準	24	48,000	526.0	22,400.0	466.7	130 nm
			高速	24	67,000	333.0	28,440.0	424.5	STMicro
		19)	標準	25	34,959	161.0	6,594.6	188.6	130 nm
					47,434	377.0	15,441.9	325.5	UMC
	1,088	14)	標準	25	56,316	487.8	21,229.1	377.0	180 nm
									UMC
		15)	標準	24	50,000	949.0	43,011.0	860.4	90 nm
					27,500	149.3	6,768.2	246.1	UMC
SHA -256	512	本論文		72	15,574	505.1	3,591.5	230.6	90 nm
					9,563	349.7	2,486.4	260.0	STMicro
					8,230	209.2	1,487.7	180.8	

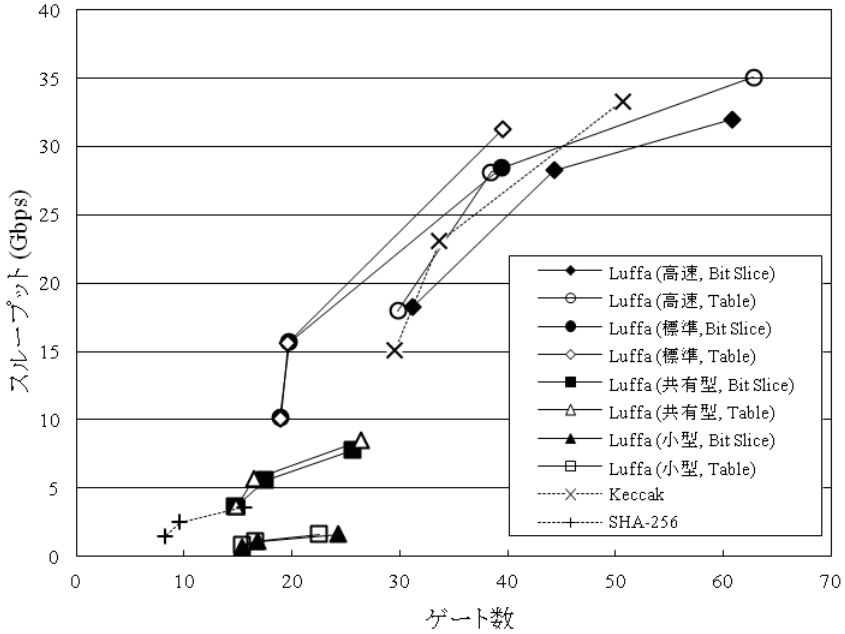

図 10　Luffa，Keccak，SHA-256 の回路性能比較（ASIC）
Fig. 10　Performances of Luffa, Keccak and SHA-256 on ASIC.

してしまった．これは Step function の演算がシンプルなため，SubCrumb や MixWord の再利用による削減量よりも，スケジューリングに必要なマルチプレクサやデマルチプレクサなどの付加回路が大きくなったためである．サイクル数は共有型と比較して単純な回路分割では約 8 倍となるところ，スケジューリングの工夫により約 5 倍に抑えられ，スループットの低下を最小限にしている．それでも，Step function 内のデータ依存関係が異なる関数ブロックでの並列処理を困難としたことにより，サイクル数は 129 と増大している．とはいえ，Keccak では関数 R 内のデータ依存関係がさらに強く，共有化による小型化では並列処理ができずにすべて逐次処理するしかなく，提案者の実装でも 5,160 サイクル要してしまっているのである[10)]．

回路効率の比較では，Table 構成の S-Box を用いた標準型の実装が 791.1 kbps/gates と最も高くなった．なお，S-box が 4 bit と小規模なことから，Bit slice と Table 構成の差はほとんど見られなかった．

表 2　Luffa，Keccak，SHA-256 の回路実装性能（Virtex-5）
Table 2　Synthesis results of Luffa, Keccak and SHA-256 on Virtex-5.

アルゴリズム	ブロックサイズ (bits)	文献	実装方式	S-box	サイクル	回路規模 (Slice)	動作周波数 (MHz)	スループット (Mbps)	回路効率 (Mbps /Slice)	デバイス
Luffa	256	本論文	高速	Bit Slice	5	1,548	137.59	7,044.5	4.55	xc5vlx 30-1
						1,256	119.11	6,098.4	4.86	
				Table	5	1,538	137.25	7,027.1	4.57	
						1,256	119.21	6,103.3	4.86	
			標準	Bit Slice	9	1,004	183.15	5,209.6	5.19	
						899	182.56	5,192.8	5.78	
				Table	9	1,004	183.15	5,209.6	5.19	
						899	182.56	5,192.8	5.78	
			共有	Bit Slice	25	891	144.56	1,480.3	1.66	
						750	129.71	1,328.2	1.77	
				Table	25	891	144.31	1,477.7	1.66	
						750	129.71	1,328.2	1.77	
			小型	Bit Slice	129	1,597	128.10	254.2	0.16	
						1,121	121.02	240.2	0.21	
				Table	129	1,597	128.10	254.2	0.16	
						1,121	121.02	240.2	0.21	
		16)	標準		9	949	340.72	9,691.6	10.21	Virtex-5
		17)	標準	Table	8	2,221	166.67	5,333.3	2.40	xc5vlx 330T-2
		20)		Table	1	9,611	48.20	12,290.0	1.28	xc5vlx 155T
			標準		9	2,303	179.60	509.0	2.21	
Keccak	1,024	本論文			24	1,641	148.19	6,322.9	3.85	xc5vlx 30-1
						1,257	130.35	5,561.7	4.43	
	1,088	16)			24	1,272	282.73	12,817.1	10.08	Virtex-5
		17)			25	1,117	189.00	8,225.3	7.36	xc5vlx 330T-2
SHA -256	512	本論文			72	416	65.34	464.6	1.12	xc5vlx 30-1
						286	46.63	331.6	1.16	

図 10 における速度優先の実装比較では，Luffa の高速型が 30〜50 Gbps 間で Keccak と重なり，スループット・回路規模ともに同等の性能となっている．一方，回路規模優先の実装で，Keccak は 29,548 gates が最小なのに対して，Luffa の標準型は 18,907 gates，共有型は 14,710 gates と半分の回路規模となり，Luffa の独立したデータパスによる関数 Q の共有の効果が大きく現れている．

高速化においては Keccak の関数 R の処理ブロックを 2〜24 個直列につないでいくことで，Luffa には実現できない高いスループットを得ることが可能と考えられる．しかしながら，SHA-256 に対して標準型は 1 桁も高速なため，小型実装も可能な Luffa の方がアプリ

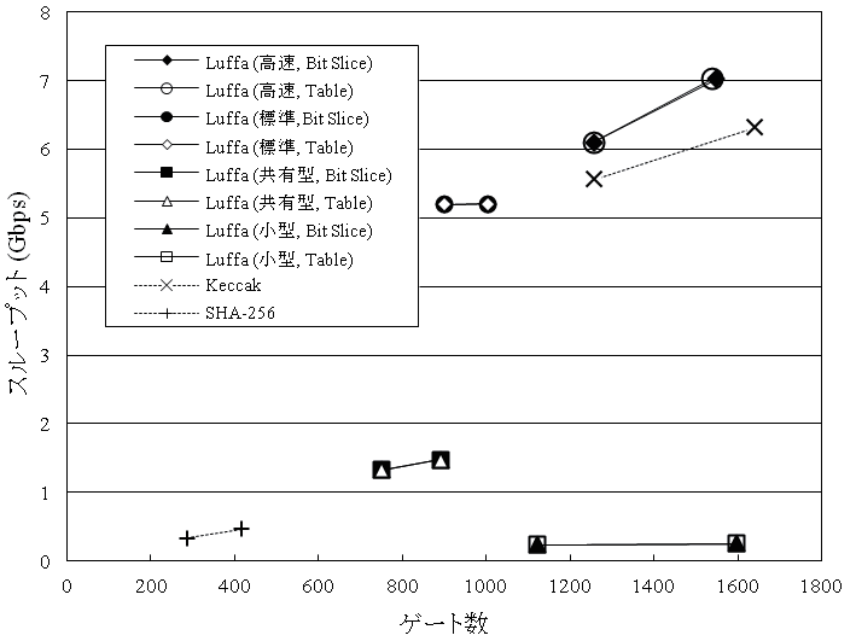

図 11　Luffa，Keccak，SHA-256 の回路性能比較（Virtex-5）
Fig. 11　Performances of Luffa, Keccak and SHA-256 on Virtex-5.

ケーションの範囲が広く実用に適しているといえよう．なお，筆者の知る限り Keccak の有効な小型アーキテクチャの提案はなされていない．

以上の性能比較の結果から，Luffa の回路はスポンジ型特有の広いデータバスにより高いスループットが実現されるだけでなく，独立なデータをパイプライン処理することで演算回路の共有による小型化が可能であり，幅広いアプリケーションの要求に応じて回路規模と処理速度が選択できる柔軟なアルゴリズムであるといえる．

4.2　FPGA 実装

FPGA による実装評価では Xilinx ISE 12.2 を用い，スタンダードセル・ライブラリと同じソースコードを Virtex-5 LX30 および Spartan-6 LX75 に対して速度優先と規模優先の 2 種類の設定で論理合成を行った．Virtex-5 の結果を**表 2** と**図 11** に，Spartan-6 の結果を**表 3** と**図 12** に示す．FPGA の回路規模は LUT（6 bit 入力 Look Up Table）と Flip-Flop

表 3　Luffa，Keccak，SHA-256 の回路実装性能（Spartan-6）
Table 3　Synthesis results of Luffa, Keccak and SHA-256 on Spartan-6.

アルゴリズム	ブロックサイズ (bits)	文献	実装方式	S-box	サイクル	回路規模 (Slice)	動作周波数 (MHz)	スループット (Mbps)	回路効率 (Mbps /Slice)
Luffa	256	本論文	高速	Bit Slice	5	1,545	108.2	5,539.8	3.59
						1,295	92.7	4,746.2	3.67
				Table	5	1,535	108.2	5,539.8	3.61
						1,295	92.7	4,746.2	3.67
			標準	Bit Slice	9	990	150.6	4,283.7	4.33
						928	150.9	4,292.3	4.63
				Table	9	990	150.6	4,283.7	4.33
						928	150.9	4,292.3	4.63
			共有	Bit Slice	25	874	170.7	1,747.7	2.00
						656	144.8	1,483.0	2.26
				Table	25	874	169.6	1,736.8	1.99
						592	126.9	1,299.8	2.20
			小型	Bit Slice	129	1,586	96.0	190.5	0.12
						1,143	94.0	186.6	0.17
				Table	129	1,586	96.0	190.5	0.12
						1,143	94.0	186.6	0.17
Keccak	1,024	本論文	標準		24	1,652	148.4	6,331.3	3.83
						1,353	105.2	4,490.2	3.32
SHA -256	512	本論文			72	346	54.7	388.7	1.13
						272	36.9	262.2	0.96

の組を 4 つ持つ Slice 単位で表している．また表 2 には，デバイスなどの条件が異なる文献 16)，17)，20) の結果を参考として掲載している．

Luffa は ASIC 実装と同様に，Virtex-5 と Spartan-6 でそれぞれ，7,044 Mbps（1,548 Slices）および 5,540 Mbps（1,535 Slices）という高スループットから，750 Slices（1,328 Mbps）および 592 Slices（1,300 Mbps）といった小型まで柔軟な実装に対応できることが分かる．しかし，Step function 内を分割する小型アーキテクチャでは，4 本の 64 bit バスを切り替えるのに FPGA では高コストなマルチプレサが必要で，これに多数の LUT が使用された結果，回路規模は高速型と同程度となりながらスループットも低下してしまった．FPGA では，ASIC のように様々なゲートを自由に選択することができず，単一の基本回路（ここでは Slice）の必要な部分だけを利用することになる．したがって，FPGA 向けの小型設計では単純な論理ブロックの削減ではなく，基本回路の構造に合った無駄のない論理となるように全体のアーキテクチャを検討する必要がある．

Keccak はスタンダードセル･ライブラリによる実装と同様に高速実装に向いており，Virtex-5 と Spartan-6 でそれぞれ，速度優先では，6,323 Mbps（1,641 Slices）および 6,331 Mbps

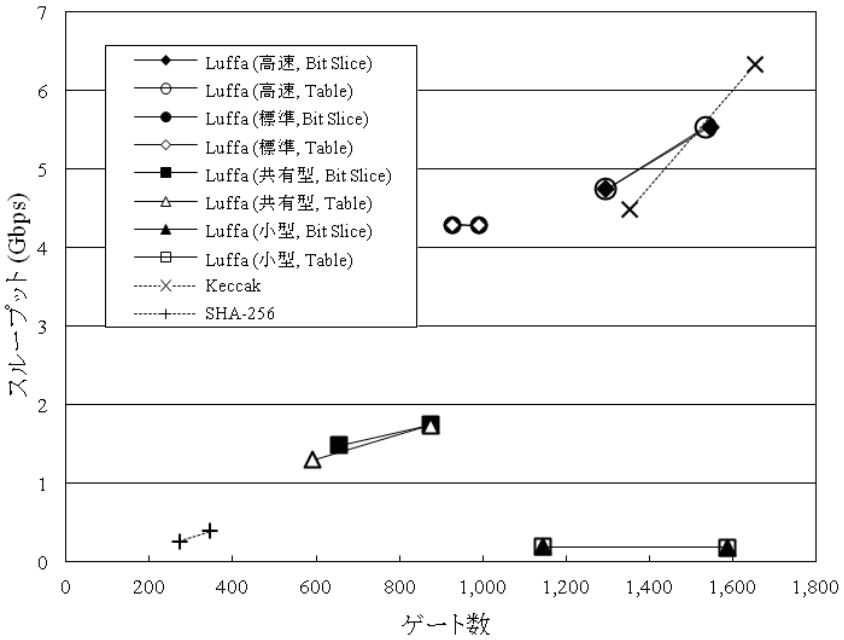

図 12　Luffa，Keccak，SHA-256 の回路性能比較（Spartan-6）
Fig. 12　Performances of Luffa, Keccak and SHA-256 on Spartan-6.

（1,652 Slices）と，Luffa の高速型と同等の性能であった．また，規模優先では 1,257 Slices（5,562 Mbps）および 1,353 Slices（4,490 Mbps）と Luffa の約 2 倍となっている．

以上の結果から，FPGA 実装では Luffa のアーキテクチャ間の速度・回路規模の差がより大きく現れるとともに，Keccak との比較では ASIC 実装と同様に小型実装における優位性が明らかとなった．

5. おわりに

本論文では，次世代ハッシュ関数 SHA-3 の候補として二次評価まで進んだアルゴリズム Luffa に対して，高速から小型まで 4 つのハードウェア・アーキテクチャを提案した．そして，それらのアーキテクチャによる回路を，90 nm CMOS スタンダードセル・ライブラリと 2 種類の FPGA デバイスを対象に論理合成し，スループット，回路規模，回路効率の性能評価を行った．また，Luffa と同じスポンジ型の SHA-3 候補である Keccak との実装性能を比較するため，アルゴリズム開発者の手によるハードウェアコードを同じ条件で論理合成した．その結果，Luffa はスタンダードセル・ライブラリと FPGA いずれにおいても，速度重視の実装で Keccak が同等のスループットと回路規模である一方，回路規模重視の実装では Keccak の約半分のサイズが実現され，小型実装での優位性が示された．

スポンジ型のハッシュ関数の Luffa は Keccak と同様にデータバス幅が広いため，SHA-256 などの従来のハッシュ関数に対してきわめて高いスループットを得ることができる．さらに Luffa は，独立したデータバスを有するため，そのバス間での演算器の共有やパイプライン化による小型化が可能となり，高速から小型の回路実装まで高い柔軟性を持つことが示された．また，Luffa と Keccak のアルゴリズムの構造の違いが回路実装性能に与える影響を考察した結果，内部バスにおける処理データブロックの独立性とスケジューリングの容易性が演算回路の共有化を可能とし，大きな回路規模低減効果をもたらすことが知見として得られた．

ハッシュ関数では，メッセージブロックの演算結果が次のメッセージブロック入力に加算されるため，各メッセージブロックを独立にパイプライン処理することができないという特徴がある．したがって，効率的な実装を行うには，1 つのメッセージブロック内で処理を分割しデータの流れが滞らないようにどうスケジューリングするかが重要となる．また，ハッシュ関数以外の暗号アルゴリズムにおいても出力データを入力側にフィードバックする動作モードが利用されることから，本論文で得られた知見はハッシュ関数以外のアルゴリズムの開発においても重要となる．ハッシュ関数を含む標準の暗号アルゴリズムは様々なアプリケーションで利用されることから，Luffa のように高速から小型まで要求性能に応じた多様な実装形態が可能な “柔軟性” が，今後のアルゴリズムの設計・評価における重要な指針となるであろう．

謝辞　本研究は JST 戦略的国際科学技術協力推進事業（共同研究型）「日本–フランス共同研究」組み込みシステムにおける暗号プロセッサの物理攻撃に対する安全性評価の一環として実施されたものである．

参 考 文 献

1) Wang, X., Yin, Y.L. and Yu, H.: Finding Collisions in the Full SHA-1, *CRYPTO 2005*, LNCS 3621, pp.17–36 (2005).
2) Wang, X. and Yu, H.: How to Break MD5 and Other Hash Functions,

EUROCRYPT 2005, LNCS 3494, pp.19–35 (2005).
3) National Institute of Standards and Technology (NIST): Cryptographic Hash Algorithm Competition,
available from ⟨http://csrc.nist.gov/groups/ST/hash/sha-3/index.html⟩.
4) ECRYPT II: THE SHA-3 Zoo,
available from ⟨http://ehash.iaik.tugraz.at/wiki/The_SHA-3_Zoo⟩.
5) Aumasson, J.-P., Henzen, L., Meier, W. and Phan, R.C.-W.: SHA-3 proposal BLAKE, available from ⟨http://www.131002.net/blake/⟩.
6) Gauravaram, P., Knudsen, L.R., Matusiewicz, K., Mendel, F., Rechberger, C., Schlaffer, M. and Thomsen, S.S.: Grøstl – a SHA-3 candidate, available from ⟨http://www.groestl.info/⟩.
7) Wu, H.: The Hash Function JH,
available from ⟨http://icsd.i2r.a-star.edu.sg/staff/hongjun/jh/⟩.
8) Bertoni, G., Daemen, J., Peeters, M. and Assche, G.V.: The Keccak sponge function family, available from ⟨http://keccak.noekeon.org/⟩.
9) Bertoni, G., Daemen, J., Peeters, M. and Assche, G.V.: Keccak sponge function family main document Version 2.1 (2010),
available from ⟨http://keccak.noekeon.org/Keccak-main-2.1.pdf⟩.
10) Bertoni, G., Daemen, J., Peeters, M. and Assche, G.V.: The Keccak reference Version 3.0 (2011),
available from ⟨http://keccak.noekeon.org/Keccak-reference-3.0.pdf⟩.
11) Bertoni, G., Daemen, J., Peeters, M. and Assche, G.V.: The Keccak SHA-3 submission Version 3 (2011),
available from ⟨http://keccak.noekeon.org/Keccak-submission-3.pdf⟩.
12) Ferguson, N., Lucks, S., Schneier, B., Whiting, D., Bellare, M., Kohno, T., Callas, J. and Walker, J.: The Skein Hash Function Family,
available from ⟨http://www.schneier.com/skein.html⟩.
13) Canniere, C.D., Sato, H. and Watanabe, D.: The Hash Function Family Luffa (Round 2), available from ⟨http://www.sdl.hitachi.co.jp/crypto/luffa/⟩.
14) Tillich, S., Feldhofer, M., Kirschbaum, M., Plos, T., Schmidt, J.-M. and Szekely, A.: High-Speed Hardware Implementations of BLAKE, Blue Midnight Wish, CubeHash, ECHO, Fugue, Grøstl, Hamsi, JH, Keccak, Luffa, Shabal, SHAvite-3, SIMD, and Skein, *IACR Eprint Report 2009/510* (2009), available from ⟨http://eprint.iacr.org/2009/510.pdf⟩.
15) Henzen, L., Gendotti, P., Guillet, P., Pargaetzi, E., Zoller, M. and Gurkaynak, F.K.: Developing a Hardware Evaluation Method for SHA-3 Candidates, *12th International Workshop on Cryptographic Hardware and Embedded Systems* (*CHES 2010*), LNCS 6225, pp.248–263 (2010).
16) Homsirikamol, E., Rogawski, M. and Gaj, K.: Comparing Hardware Performance of Fourteen Round Two SHA-3 Candidates Using *FPGAs, IACR Eprint Report 2010/445* (2010), available from ⟨http://eprint.iacr.org/2010/445.pdf⟩.
17) Baldwin, B., Hanley, N., Hamilton, M., Lu, L., Byrne, A., O'Neill, M. and Marnane, W.P.: FPGA Implementations of the Round Two SHA-3 Candidates, *The 2nd SHA-3 Candidate Conference* (2010), available from ⟨http://csrc.nist.gov/groups/ST/hash/sha-3/Round2/Aug2010/documents/papers/BALDWIN_FPGA_SHA3.pdf⟩.
18) Akin, A., Aysu, A., Ulusel, O.C. and Savas, E.: Efficient Hardware Implementations of High Throughput SHA-3 Candidates Keccak, Luffa and Blue Midnight Wish for Single- and Multi-Message Hashing, *The 2nd SHA-3 Candidate Conference* (2010), available from ⟨http://csrc.nist.gov/groups/ST/hash/sha-3/Round2/Aug2010/documents/papers/SAVAS_SHA3_NIST_final.pdf⟩.
19) Guo, X., Huang, S., Nazhandali, L. and Schaumont, P.: Fair and Comprehensive Performance Evaluation of 14 Second Round SHA-3 ASIC Implementations, *The 2nd SHA-3 Candidate Conference* (2010), available from ⟨http://csrc.nist.gov/groups/ST/hash/sha-3/Round2/Aug2010/documents/papers/SCHAUMONT_SHA3.pdf⟩.
20) Ramakers, W. and Narinx, H.: Implementation and evaluation of SHA-3 candidates on FPGA (Duch), available from ⟨http://ehash.iaik.tugraz.at/uploads/6/62/Ramakers_Narinx2010ECHO-Hamsi-Luffa_Thesis_DUTCH.pdf⟩, available from ⟨http://ehash.iaik.tugraz.at/uploads/1/12/Ramakers_Narinx2010ECHO-Hamsi-Luffa_ExtendedAbstract_ENGLISH.pdf⟩.
21) Knezevic, M. and Verbauwhede, I.: Hardware Evaluation of the Luffa Hash Family, *4th Workshop on Embedded Systems Security* (*WESS 2009*), available from ⟨http://www.cosic.esat.kuleuven.be/publications/article-1282.pdf⟩.
22) Matsuo, S., Knezevic, M., Schaumont, P., Verbauwhede, I., Satoh, A., Sakiyama, K. and Ota, K.: How Can We Conduct "Fair and Consistent" Hardware Evaluation for SHA-3 Candidate?, *The 2nd SHA-3 Candidate Conference* (2010),
available from ⟨http://csrc.nist.gov/groups/ST/hash/sha-3/Round2/Aug2010/documents/papers/MATSUO_SHA-3_Criteria_Hardware_revised.pdf⟩.
23) SASEBO (Side-channel Attack Standard Evaluation BOard): SHA-3 Hardware, available from ⟨http://staff.aist.go.jp/akashi.satoh/SASEBO/en/sha3/implement.html⟩.
24) The National Institute of Standards and Technology (NIST): Specification for the Advanced Encryption Standard (AES), *FIPS 197* (2001), available from ⟨http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf⟩.
25) CMP (Circuit Multi-Projects): CMOS 90nm (CMOS090) from STMicroelectron-

ics, available from ⟨http://cmp.imag.fr/products/ic/?p=STCMOS090⟩.
26) Xilinx: Virtex-5 FPGA family, http://www.xilinx.com/products/virtex5/
27) Xilinx: Spartan-6 FPGA family,
available from ⟨http://www.xilinx.com/products/spartan6/⟩.
28) The National Institute of Standards and Technology (NIST): Specification for the Secure Hash Standard, *FIPS 180-2* (2002),
available from ⟨http://csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf⟩.
29) Satoh, A. and Inoue, T.: ASIC-hardware-focused comparison for hash functions MD5, RIPEMD-160, and SHA, Integration, *the VLSI Journal, Special issue: Embedded cryptographic hardware*, Vol.40, Issue 1, pp.3–10 (2007).

(平成 23 年 2 月 16 日受付)
(平成 23 年 9 月 12 日採録)

推　薦　文

明確な分析を行っており，論文全体にわたり高いレベルにある研究である．Luffa（高速），Luffa（標準），Luffa（小型 1），Luffa（小型 2），Keccak，SHA-256 それぞれについて，ゲート数を変化させた ASIC 回路実装性能や回路性能比較を与えており，ゲート数と実装の関係を示す有益な情報を提供している．安全性の高い SHA-3 が誕生するために，本技術による貢献を期待する．　（コンピュータセキュリティ研究会主査　菊池浩明）

片下　敏宏

2006 年筑波大学大学院システム情報工学研究科修了．同年（独）産業技術総合研究所特別研究員．2008 年産業技術総合研究所任期付研究員，現在に至る．高速演算，セキュリティに関するハードウェアおよびソフトウェア設計の研究に従事．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）．

菅原　　健

2006 年東北大学工学部情報工学科卒業．2008 年同大学大学院情報科学研究科修士課程修了．2011 年同博士課程修了．2008〜2011 年日本学術振興会特別研究員を兼任．2011 年より三菱電機株式会社に勤務，現在に至る．高性能暗号ハードウェアの設計およびサイドチャネル攻撃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 パリテック客員研究員を兼任．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）．