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

2008－CSEC－42（30）
2008 / 7 / 25

再構成・拡張可能なプロセッサへのブロック暗号Camelliaの実装

松尾 一慶† 阿部 公輝†

† 電気通信大学大学院 情報工学専攻
〒 182-8585 東京都調布市調布ヶ丘 1-5-1
E-mail: †{kazu,abe}@cacao.cs.uec.ac.jp

あらまし ブロック暗号アルゴリズム Camellia を再構成・拡張可能なプロセッサに実装し，構成パラメータを変えること，および，命令を拡張することによる所要サイクル数，総消費電力の変化をシミュレーション実験により測定した．キャッシュに着目した再構成により，所要サイクル数で約 10%削減された．主要な処理を 1 命令で実行する命令拡張により，所要サイクル数は拡張前の 1/106，総消費電力値は拡張前の 1/18 となった．命令拡張が与える影響を AES と比較すると，Camellia では総消費電力の削減に大きな効果がある．

キーワード Camellia，ブロック暗号，再構成・拡張可能プロセッサ，消費電力，サイクル数

Implementation of Block Cipher Camellia with a Configurable and Extensible Processor

Kazuyoshi MATSUO† and Kôki ABE†

† Department of Computer Science, The University of Electro-Communications
1-5-1 Chofugaoka, Chofu-shi, Tokyo 182-8585 Japan
E-mail: †{kazu,abe}@cacao.cs.uec.ac.jp

Abstract Camellia, a block cipher algorithm, has been implemented on a configurable and extensible processor to measure cycles and power consumption required for processing the algorithm when varying the configuration parameters and/or extending the instruction set of the processor. Results of optimizing configuration parameters revealed the reduction of processing cycles by approximately 10%. Extending the instruction set reduced processing cycles and total power consumption by 1/106 and 1/18, respectively. The extension has larger effect on reducing total power consumption for processing Camellia than AES.

Key words Camellia, block cipher, configurable and extensible processor, power consumption, processing cycle

1. はじめに

ネットワーク上でのプライバシー確保，安全性確保のために，暗号技術は重要なものとなっている．組み込み分野においても暗号化実装が行われるようになってきており，暗号，特に広範囲で用いられる共通鍵暗号，を効率良く実装することは重要である．

計算量の多い処理を高速かつ低コストで実装する手段の一つとして，特定用途向け命令セットプロセッサ（Application Specific Instruction Set Processor, ASIP）がある．ASIP は特定用途向け IC（Application Specific Integrated Circuit, ASIC）の高性能性と汎用プロセッサの柔軟性の特徴を併せ持つ．さらに，開発期間の短縮の面でも有利である．ASIP を設計する効率的な方法として，再構成・拡張可能なプロセッサを用いる手法があり，たとえば，セキュリティプロトコル IPsec の実装例がある [1], [2].

再構成可能とは，基本的なパラメータ（レジスタファイルの数，最大命令長，キャッシュサイズ，バス幅，バイト順序，パイプライン段数等）の変更が可能であることを言う．拡張可能とは，命令，拡張ユニット，I/O 等を開発者がアプリケーションに合わせて独自に追加することができることを言う．このように再構成・拡張可能な性質は，高速，低コスト，低電力な ASIP を開発する上で有用である．

AES と同等以上の安全性を持つ共通鍵暗号として，Camellia がある [3]～[5]．これまで，Camellia に対し，一般的な汎用プロセッサ上でのソフトウェア実装の評価 [6] や，ASIC 設計によるハードウェア実装の評価 [7] はなされてきたが，再構成・拡張可能なプロセッサへの実装と評価は行われていない．AES

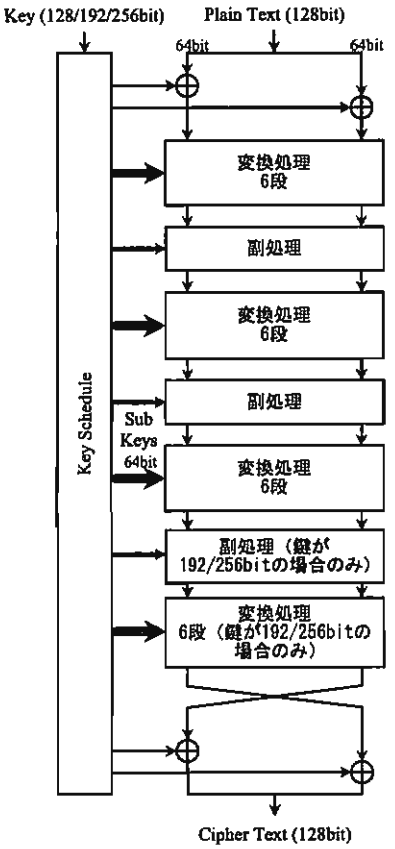

図 1　Camellia の暗号処理

Fig. 1　Encryption of Camellia.

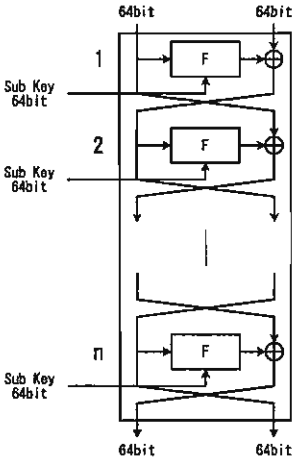

図 2　変換処理（n 段の Feistel 構造）

Fig. 2　n-round Feistel structure.

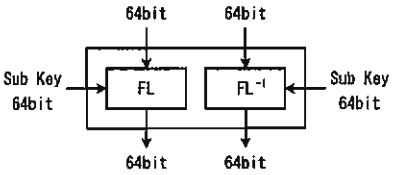

図 3　副　処　理

Fig. 3　FL/FL^{-1} functions.

は繰り返し構造に SPN 構造を用いているのに対し，Camellia では Feistel 構造を用いている．AES の繰り返し構造は Key Addition, Substitution, Row Shifting, Column Mixing と呼ばれる処理からなる．一方 Camellia は単純な置換と換字処理からなる．このような構造の違いが再構成・拡張可能なプロセッサにおける性能やコストにどのような影響を与えるかを調べることは意味がある．さらに，再構成・拡張可能プロセッサによる暗号処理実装を消費電力について定量的に評価された例はこれまでない．

そこで，本研究では再構成・拡張可能なプロセッサ上に Camellia を実装し，性能と電力の面から従来法と比較，評価する．さらに，AES の実装との比較検討も行う．

2.　共通鍵ブロック暗号 Camellia

Camellia は，日本電信電話株式会社（NTT）と三菱電機株式会社が共同で開発した国産の 128 ビット共通鍵ブロック暗号アルゴリズムである [3]〜[5]．基本特許は無償化され，オープンソース化されている．標準化作業が進行し，各種国際標準暗号にも認定されている．

Camellia の暗号処理の概要を図 1 に示す．Camellia の入出力インターフェースは AES 互換であり，ブロック長 128 ビット，鍵長は 128, 192, 256 ビットの 3 種類から選択できる．図 1 の変換処理は図 2 に示す Feistel 構造による．鍵長によって，変換処理の Feistel 構造の段数の合計値が異なる（表 1）．Feistel 構造 1 段は次式の処理を行う．

$$\begin{aligned} L_r &= R_{r-1} \oplus F(L_{r-1}, K_r) \\ R_r &= L_{r-1} \end{aligned} \tag{1}$$

ここで，関数 F は，Feistel 構造における F 関数であり，8 ビットから 8 ビットへの換字を行う S-Box8 個と 8 ビットごとの排他的論理和等の処理から構成される．添字 r はラウンド数を，L_r，R_r はラウンド r における Feistel 構造への左，右出力を表す．K_r は鍵拡張処理（Key Schedule）によって生成されるラウンド r における副鍵である．

また，構造の規則性を崩すため図 3 のような副処理を挿入する．図中の FL，FL^{-1} 関数は副鍵との論理演算，循環シフト，排他的論理輪から成る．

また，図 1 の中の Key Schedule は秘密鍵 K から，いくつかの 64 ビット長の副鍵を生成する．副鍵は入力直後に 2 つ，出

表 1　Camellia の鍵長と Feistel 構造の段数の関係

Table 1　Key length and Feistel structure rounds of Camellia.

	鍵長 [bit]	ブロック長 [bit]	段（ラウンド）数
Camellia128	128	128	18
Camellia192	192	128	24
Camellia256	256	128	24

表 2　Camellia の実装の種類
Table 2　Implementations of Camellia.

	ソフトウェアコード	プロセッサ	
		再構成	命令拡張
Camellia Base 実装	Fundamental	×	×
Camellia 実装 A1	Fundamental	○	×
Camellia 実装 A2	GPL	○	×
Camellia 実装 B	Extended	○	○

表 3　AES の実装の種類
Table 3　Implementations of AES.

	ソフトウェアコード	プロセッサ	
		再構成	命令拡張
AES Base 実装	Fundamental	×	×
AES 実装 A	Fundamental	○	×
AES 実装 B	Extended	○	○

力直前に 2 つ，変換処理の各ラウンドに 1 つ，各副処理に 2 つ使用される．たとえば，鍵長が 128 ビットの場合は 26 個の副鍵を生成する．

復号化は，副鍵の適用順序を暗号化と逆順にすることにより，暗号化と同様の手順で処理される．

3. 実験条件

本研究では，再構成・拡張可能なプロセッサとして，米 Tensilica 社の Xtensa LX2 プロセッサ [8] を用いる．実装とその評価には，Xtensa LX2 の Xtensa 命令セットシミュレータ（ISS）（Ver. 7.1.0.2）を使用する．ISS は Windows XP 上で実行する．

Camellia と AES の実装の種類を表 2 と表 3 に示す．ソフトウェアコードは System C による記述である．Fundamental はアルゴリズムに忠実な素朴なコード，Camellia の GPL は NTT によるオープンソースコードである [5]．GPL は，副鍵の生成手順や，換字置換の方法等に工夫がなされ，ソフトウェア的な高速化手法が施されたコードである．Extended は，Fundamental コードを基に拡張命令を付加した命令セットを利用したコードである．

表 2，表 3 において，再構成欄の○（×）印はそのソフトウェアコードに対してプロセッサの構成パラメータを変えて（変えずに）実装することを表す．Camellia の Fundamental コード，GPL コードに対し最適な構成パラメータを持つ実装を，それぞれ Camellia 実装 A1，Camellia 実装 A2 と呼ぶ．AES の Fundamental コードに対し最適な構成パラメータを持つ実装を AES 実装 A と呼ぶ．

表 2，表 3 において，命令拡張欄の○（×）印はそのソフトウェアコードに対して命令拡張を行う（行わない）ことを表す．Camellia，AES に対し命令拡張を行った後，構成パラメータを最適化した実装をそれぞれ Camellia 実装 B，AES 実装 B と呼ぶ．

再構成，命令拡張をしないプロセッサ上で，Fundamental コードを実行する実装を Base 実装と呼ぶ．

4. 再構成の実験

Camellia と AES の暗号化，復号化の処理の複雑さはソフトウェア実行に要するサイクル数に反映される．ソフトウェア実行に要するサイクル数はプログラムを実行するプロセッサの構成を変えると変わる．本研究で用いる Xtensa は，レジスタファイルの数，最大命令長，キャッシュサイズ，バス幅，バイト順序，パイプライン段数等のパラメータを変更できる．これらのパラメータを変化させたときのサイクル数の変化はアルゴリズムの性質を反映すると考えられる．ブロック暗号処理では，メモリとの間のデータのやり取り等が多く行われるので，この実験ではキャッシュに着目したプロセッサの再構成を行う．

4.1 実験方法

Camellia の Fundamental コード，GPL コードに対し，データキャッシュと命令キャッシュのそれぞれについて，容量，ラインサイズ，連想度を変化させたときのサイクル数の変化を測定する．これらのパラメータの変化可能な範囲は図 5 に示す．なお，この実験ではレジスタファイルの数は 32，最大命令長は 64 ビット，データバス幅は Camellia と AES が扱うデータ単位に合わせ 128 ビット，バイト順序は GPL コードに合わせビッグエンディアン，パイプライン段数は 5 段とする．データキャッシュ，命令キャッシュともに容量 4KB，ラインサイズ 16 バイト，連想度 1 をベース設定とし，データキャッシュまたは命令キャッシュのある 1 つのパラメータを変化させる．その他のパラメータはツールのデフォルト値とする．測定結果から，処理に要するサイクル数をできるだけ少なくし，かつ，プロセッサのコストを少なくするような最適なパラメータを求める．

AES の Fundamental コードに対しても同様に実験を行い，両者の処理の複雑さと性質について比較考察する．なお，Camellia と AES の Base 実装（Fundamental コードを用い，再構成，命令拡張をしない実装）の構成パラメータは図 5 の最小値とする．

4.2 実験結果と考察

Camellia と AES の暗号化処理のサイクル数を測定した結果を図 4 に示す．

Fundamental コードについては，図 4 から，どのようなパラメータにおいても AES の方が Camellia よりもサイクル数が多い．処理の複雑さはこの順となると考えられる．Camellia のコードについては，前述のようにソフトウェア的な高速化手法が施されている GPL コードは Fundamental コードよりもサイクル数が少ない．以下では，Camellia と AES の Fundamental コードに対する実験結果について述べる．

データキャッシュ容量を増加させた場合はいずれの処理もサイクル数は減少し，いずれも約 4KB で飽和する．Camellia においては，256 バイトの S-box を 4 つ（1KB），副鍵の格納に約 0.5KB 使用する．この他に各ラウンド処理手続きの呼び出しとメインルーチンへの戻りのためのメモリ領域が必要となる．

命令キャッシュ容量についても同様であるが，サイクル数の減少の度合いはデータキャッシュの場合よりも大きい．実行ファイルサイズの差から，処理に要する部分のコードサイズを推定すると，Camellia の場合約 3.8KB，AES の場合約 2.5KB で

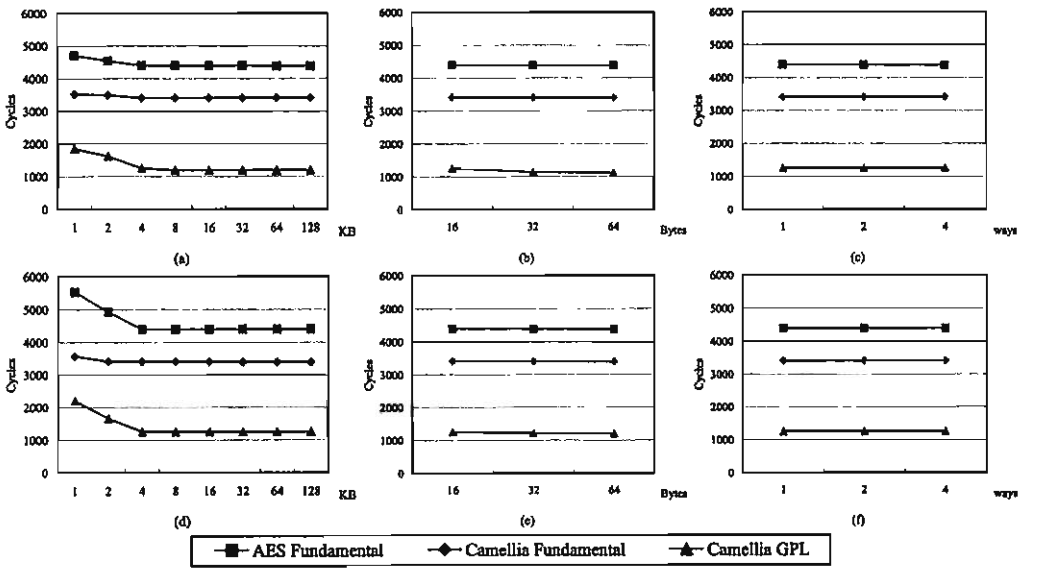

図 4 構成パラメータを変えたときの所要サイクル数
(a) データキャッシュ容量, (b) データキャッシュラインサイズ, (c) データキャッシュ連想度, (d) 命令キャッシュ容量, (e) 命令キャッシュラインサイズ, (f) 命令キャッシュ連想度

Fig. 4 Processing cycles when varying configuration parameters: (a) Data cache size; (b) Data cache line size; (c) Data cache associativity; (d) Instruction cache size; (e) Instruction cache line size; (f) Instruction cache associativity.

ある．これより，命令キャッシュ容量の効果は 4KB で飽和したものと考えられる．

データキャッシュ，命令キャッシュそれぞれのラインサイズ，連想度を変化させてもサイクル数の大きな変化は見られない．ラインサイズの大きさは空間的局所性の範囲の程度を反映している．どちらのアルゴリズムもこの範囲は小さいことを示す．連想度の大きさは置き換え競合の程度に影響を与える．どちらのアルゴリズムもこの競合はあまりおこらないことを示す．

以上の実験結果から最適なパラメータを求めた結果を図 5 に示す．このパラメータによる Camellia の実装を A1，AES の実装を A とする．図 5 から，AES の暗号化処理を除き，Camellia と AES に対する最適なパラメータの値はよく似ている．これは，Camellia と AES のアルゴリズムの性質が似ていることを示す．

GPL コードに対する最適なパラメータによる実装 A2 のパラメータも図 5 に示してある．

5. 命令拡張の実験

Xtensa LX2 プロセッサを用いて Camellia の鍵拡張・暗号化・復号化処理に対し，専用命令を追加する．命令拡張が Camellia の暗号化，復号化のサイクル数と消費電力に与える効果を調べる．再構成がこれらの処理の消費電力に与える効果も調べ，サイクル数に与える効果と併せ，AES の場合と比較する．

5.1 鍵拡張・暗号化・復号化処理回路

各ラウンドの副鍵を生成するための前処理を行う回路（鍵拡張前処理回路）を図 6 に示す．また，暗号化・復号化処理回路を図 7 に示す．鍵拡張前処理は暗号化・復号化処理に先立って予め行う．副鍵は鍵拡張前処理の結果を用いて暗号・復号処理の専用命令内で動的に生成する．

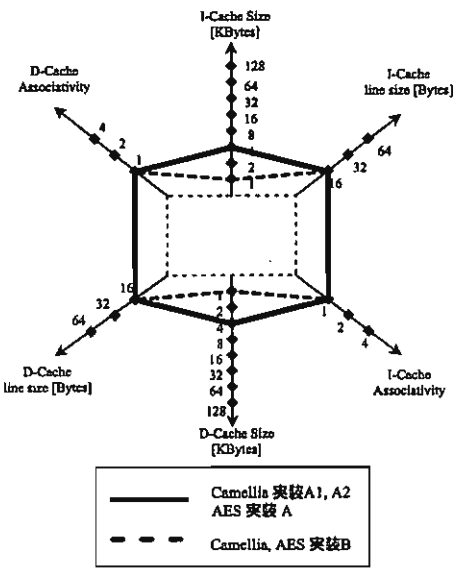

図 5 最適な再構成パラメータ

Fig. 5 Optimized configuration parameters.

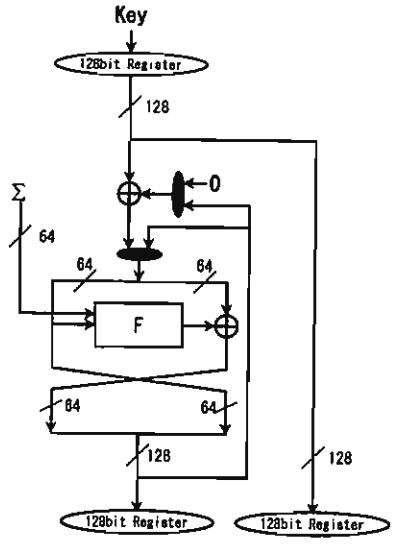

図 6 鍵拡張前処理回路

Fig. 6 Pre-processing circuits for key extension.

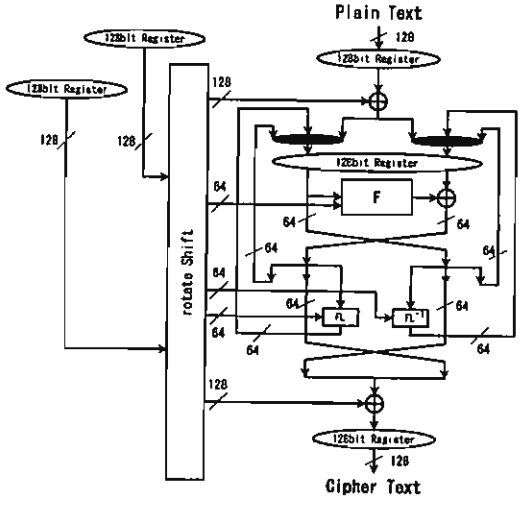

図 7 暗号化・復号化処理回路

Fig. 7 Encryption/decryption circuits.

暗号化・復号化回路は，式 (1) を主とする処理を必要なラウンド数分繰り返すことにより，暗号化処理を 1 命令で実行できる．復号化は副鍵の適用順序が異なる（逆順）が，暗号化と同一の回路を利用して実現できる．換字表は回路内に用意されている．

データ転送の効率化を図るため，データバス幅は 128 ビットとし，128 ビットレジスタを 4 つ実装した．

表 2 における Camellia の Extended コードは命令拡張されたプロセッサを用いて暗号化，復号化処理を行うコードである．Extended コードに対して最適な構成パラメータを持つ実装を B と呼ぶ．Camellia 実装 B のパラメータも図 5 に示してある．命令拡張を行った場合，プログラムコード量，データ量ともに

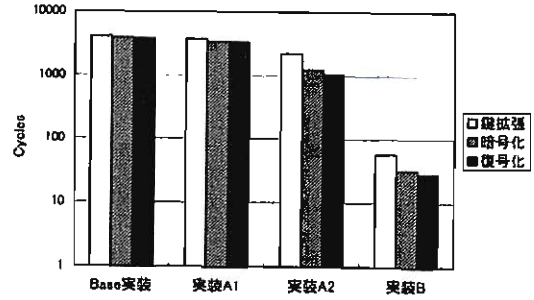

図 8 Camellia の所要サイクル数

Fig. 8 Processing cycles for Camellia.

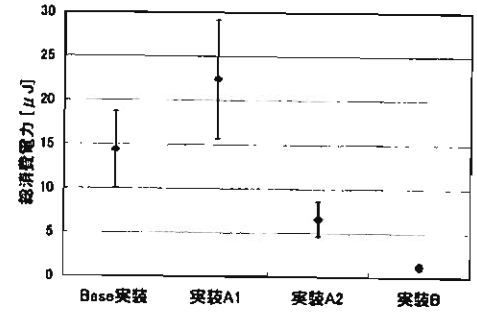

図 9 Camellia の暗号化処理の総消費電力値

Fig. 9 Total power consumption for Camellia encryption.

大幅に減少するので，最適なキャッシュの構成パラメータは，図に示すように最小値をとる．

なお，AES の Extended コードについても再構成の実験の結果は Camellia の場合と同様である．（最適なキャッシュの構成パラメータは最小値をとる．）AES 実装 B のパラメータも図 5 に示してある．

5.2 結果と考察

性能の評価は，1 ブロック分の暗号化・復号化に要するサイクル数と総消費電力値を計測することにより行う．

各実装の所要サイクル数と総消費電力値をそれぞれ，図 8，図 9 に示す．図において，Base 実装，実装 A1，実装 A2，実装 B の意味は表 2 に示すとおりである．図 8 では，鍵拡張前処理，暗号化，復号化に要するサイクル数を分けて示してある．図 9 では，使用ツールの測定誤差 30%を示してある．

再構成を行わない Base 実装と比較して，再構成を行った実装 A1 では，約 10%のサイクル数が削減された．Fundamental コードに対し再構成を行った実装 A1 と比較して，GPL コードに対し再構成を行った実装 A2 では，約 63%のサイクル数が削減された．Base 実装と実装 A1 の比較では，電力の明らかな低下は認められないが，これは，実装 A1 ではサイクル数が約 10%しか削減されていないにもかかわらず，キャッシュ容量が増えたため，その分消費電力が増えたことによると考えられる．

命令拡張によるサイクル数削減への効果は極めて大きい．実装 B の暗号化処理は，実装 A1 のサイクル数の 1/106 となった．総消費電力値削減への効果も大きい．実装 A1 と比較して，

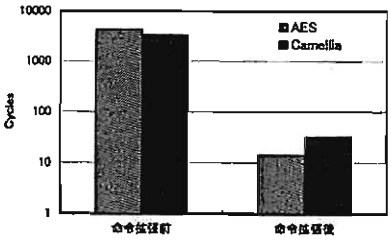

図 10 AES と Camellia の命令拡張前後の暗号化処理の所要サイクル数（AES の命令拡張前は実装 A，Camellia の命令拡張前は実装 A1 を表す）

Fig. 10 Processing cycles for Camellia and AES encryption with/without instruction extension. (Implementations A and A1 were evaluated for Camellia and AES without instruction extension.)

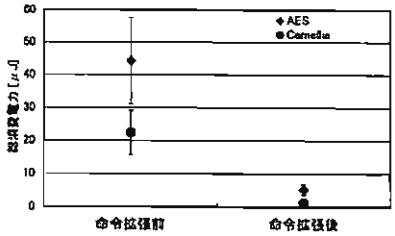

図 11 AES と Camellia の命令拡張前後の暗号化処理の総消費電力値（AES の命令拡張前は実装 A，Camellia の命令拡張前は実装 A1 を表す）

Fig. 11 Total power consumption for Camellia and AES encryption with/without instruction extension. (Implementations A and A1 were evaluated for Camellia and AES without instruction extension.)

実装 B では，（誤差を無視して）消費電力は 1/18 となった．誤差を考慮すると最低でも 1/10 の電力削減となる．鍵拡張前処理についても，1/65 のサイクル数が削減されているので，特に鍵が頻繁に更新されるような状況では有用である．

Base 実装，実装 A1，実装 A2 のプロセッサは，約 350MHz で動作する．これに対して，実装 B のプロセッサは約 300MHz で動作する．命令拡張による回路の複雑さはクリティカルパスにさほど大きな影響は与えないことを示す．

AES と Camellia の命令拡張前後の暗号化処理の所要サイクル数と総消費電力値を図 10 と図 11 に示す．AES の命令拡張前は実装 A，Camellia の命令拡張前は実装 A1 の値を示してある．AES では実装 A の暗号化処理は，実装 B のサイクル数の 1/311 となった．また，消費電力値は，（誤差を無視して）1/9 となった．誤差を考慮すると最低でも 1/5 の電力削減となる．Camellia よりも AES のほうがアルゴリズムは複雑であるので，命令拡張前の実装では，Camellia よりも AES のほうが処理に要するサイクル数は大きい．これらのアルゴリズムを処理するプロセッサが同じであれば総消費電力値は所要サイクル数に比例する．そのため，命令拡張前は Camellia よりも AES のほうが総消費電力値は大きい．命令拡張後はどちらのアルゴリズムも主要な処理が 1 命令で実行されるため，所要サイクル数は同程度となる．拡張された命令を実行する回路はアルゴリズムの複雑さを反映し，Camellia よりも AES のほうが複雑であり，消費電力も大きい．以上の理由で命令拡張が与える影響を AES と比較すると，Camellia ではサイクル数より総消費電力のほうがより大きな影響を受けると考えられる．

命令拡張によるクロック周期の増加について考察する．Camellia よりも AES のほうが拡張命令を実行する回路は複雑であるので，回路のクリティカルパスは AES のほうが長い．従って所要サイクル数で言えば命令拡張後の Camellia と AES は同程度であるが命令スループットは Camellia のほうが大きい．ただし，適切にパイプライン化すれば回路へのフィードバックが無ければ命令拡張後もスループットを同程度にすることはできる．

6. おわりに

共通鍵暗号アルゴリズム Camellia を再構成・拡張可能なプロセッサに実装し，構成パラメータを変化させること，および，命令を拡張することによる所要サイクル数，総消費電力の変化を測定した．その結果，再構成により所要サイクル数で約 10%削減された．命令拡張により所要サイクル数は拡張前の 1/106，総消費電力値は拡張前の 1/18 となった．命令拡張が与える影響を AES と比較すると，Camellia では総消費電力の削減に大きな効果がある．

謝辞 本研究を行うにあたり Xtensa 命令セットシミュレータを使用させていただいたテンシリカ株式会社に感謝する．本研究は，一部，日本学術振興会科学研究費補助金（基盤研究 (C)18500048）による．ここに記して謝意を表す．

文　献

[1] N. Potlapally, S. Ravi, A. Raghunathan, R. B. Lee, and N. K. Jha, "Configuration and Extension of Embedded Processors to Optimize IPSec Protocol Execution," IEEE Trans. VLSI Systems, Vol.15, No.5, pp.605–609, 2007.

[2] N. Potlapally, S. Ravi, A. Raghunathan, R. B. Lee, and N. K. Jha, "Impact of configurability and extensibility on IPsec protocol execution on embedded processors," Proc. Int. Conf. VLSI Design, pp.299–304, 2006.

[3] K. Aoki, T. Ichikawa, M. Kanda, M. Matsui, S. Moriai, J. Nakajima, and T. Tokita, "Camellia - A 128-bit Block Cipher," Technical report of IEICE, ISEC, Vol.100, No.76, pp.47–75, 2000.

[4] M. Matsui, J. Nakajima, and S. Moriai, "A Description of the Camellia Encryption Algorithm," RFC3713, 2004.

[5] 日本電信電話株式会社　情報流通プラットホーム研究所　情報セキュリティプロジェクト，"Camellia," http://info.isl.ntt.co.jp/crypt/camellia/index.html

[6] 中島 純子，松井 充，"Core2 上でのブロック暗号アルゴリズムの実装性能評価について，" 電子情報通信学会技術研究報告，Vol.106, No.597, pp.105–110, 2007.

[7] H. Cheng and H. M. Heys, "Compact Hardware Implementation of the Block Cipher Camellia with Concurrent Error Detection," Proc. IEEE Canadian Conf. Electrical and Computer Engineering, pp.1129–1132, 2007.

[8] Tensilica, Inc., "Xtensa LX2 Product Brief," http://www.tensilica.com/pdf/xtensa_LX2_April07.pdf