研究速報

AES 暗号と Camellia 暗号に対する暗号鍵を固定したハードウェア特殊化回路

松岡　俊佑[†a)]（正員）　　日野　善規[††]
市川　周一[†††]（正員：シニア会員）

Hardware Specialization for Key Specific AES and Camellia Cipher Circuit

Shunsuke MATSUOKA[†a)], *Member*,
Yoshiki HINO[††], *Nonmember*,
and Shuichi ICHIKAWA[†††], *Senior Member*

† 旭川工業高等専門学校，旭川市
Asahikawa National College of Technology, Asahikawa-shi, 071–8142 Japan
†† 中部電力株式会社，名古屋市
CHUBU Electric Power Co., Inc., Nagoya-shi, 461–8680 Japan
††† 豊橋技術科学大学，豊橋市
Toyohashi University of Technology, Toyohashi-shi, 441–8580 Japan
a) E-mail: matsuoka@asahikawa-nct.ac.jp

あらまし　AES 暗号と Camellia 暗号について，暗号鍵を定数に固定した回路を設計し，FPGA による実装評価を行った．その結果，従来回路と比較して論理規模が削減され，最大動作周波数が改善された．

キーワード　AES 暗号，Camellia 暗号，FPGA，ハードウェア特殊化

1. まえがき

ソフトウェアプログラミングにおいて，実行する前から一部の入力値が既知である場合，その値を用いて実行前にあらかじめプログラムコードを最適化しておくことで，実行時間を短縮することができる．こうしたプログラミング技術のことを部分評価 (Partial Evaluation)，あるいはプログラム特殊化 (Program Specializer) という [1], [2]．ハードウェア回路に対しても同様な手法が適用できる．論理回路においても入力の一部が一定値であるならば，部分評価により内部の回路を最適化することで回路規模が削減でき，動作速度が改善できる（ハードウェア特殊化）．ただし，ソフトウェアと異なりハードウェア回路では内部回路が容易に変更できないため，特殊化技術を適用した場合，ある入力値に対してのみの専用回路となり，異なる入力値に対しては別の回路を新たに生成しなければならない．したがって，ハードウェア特殊化回路を実装するには，FPGA に代表されるような再構成可能論理デバイスが適している．

本研究では，共通鍵暗号に対してハードウェア特殊化技術を適用した暗号回路を設計し，FPGA による実装評価を行う．共通鍵暗号には様々な暗号アルゴリズムがあるが，米国政府標準暗号の DES や AES が最も広く利用されている．日本国産暗号では電子政府推奨暗号 (CRYPTREC) や，ISO/IEC 国際標準暗号，インターネット標準暗号 (IETF) に選定されている Camellia 暗号がある．

本研究の目的は，Camellia 暗号回路について暗号鍵を一定値に固定したハードウェア特殊化回路を評価することである．AES 暗号や DES 暗号のハードウェア特殊化回路は先行研究 [3], [4] にて提案されているが，本研究では先行研究と比較するため AES 暗号回路を再設計した．更に AES と Camellia のハードウェア特殊化回路を Xilinx FPGA で定量的に評価した．

2. Camellia 暗号アルゴリズム

Camellia 暗号は，2000 年に NTT と三菱電機株式会社によって開発された共通鍵暗号アルゴリズムである．ブロック長は，128 bit，192 bit，256 bit から選択できる．ここでは鍵長が 128 bit の場合の Camellia 暗号アルゴリズムについて述べる．なお詳細については文献 [5] を参照されたい．

Camellia のデータ処理部は，Feistel 構造を採用しており，128 bit のデータを 64 bit ずつ分割し，左右にデータを入れ換えながら全 18 ラウンドの処理を繰り返していく．第 1 段目のラウンド処理の前には排他的論理和処理 (Pre-Whitening)，及び最終ラウンド処理の後には排他的論理和 (Post-Whitening) が実行される．また，6 ラウンドと 12 ラウンドの後には FL^{+}/FL^{-} 関数が実行される．ラウンド処理の内部は F 関数と排他的論理和から構成されている．F 関数は次式で定義

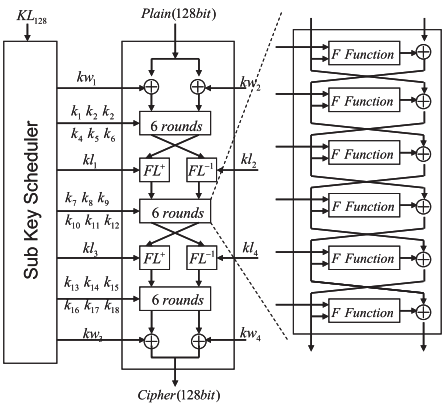

図 1　Camellia の暗号化処理過程（鍵長 128 bit）
Fig. 1　Encryption process of Camellia for 128 bit key.

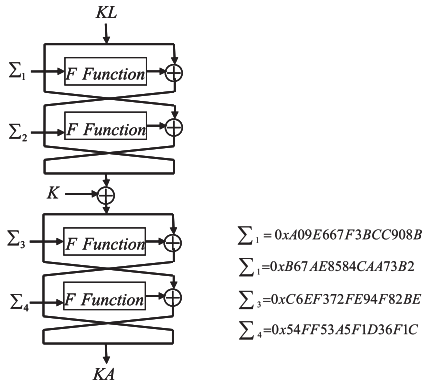

図 2　Camellia の副鍵生成過程（鍵長 128 bit）
Fig. 2　Sub key generating process of Camellia for 128 bit key.

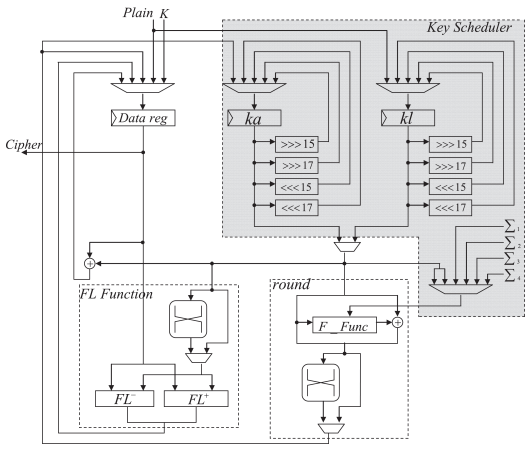

図 3　ループ型 Camellia 暗号回路
Fig. 3　Camellia encryption circuit by loop architecture.

される．

$$Y_{(64)} = P\left(S\left(X_{(64)} \oplus k_{(64)}\right)\right) \qquad (1)$$

上式の k は鍵生成部で暗号鍵をもとに生成される 64 bit の副鍵を示している．P 関数は副鍵との排他的論理和処理と，8 ビット入力 8 ビット出力の非線形変換である八つの S 関数からなる．

鍵生成部ではデータ処理部と同じくラウンド処理を繰り返していく（図 2）．図中の KL は入力暗号鍵を示している．F 関数には 128 bit の定数 (Σ_1〜Σ_4) を入力する．生成した中間鍵 KA を 15 bit，あるいは 17 bit ローテーションシフトすることで全 26 個の副鍵 (kw_1〜kw_4，kl_1〜kl_4，k_1〜k_{18}) が生成される．

3. Camellia 暗号回路

共通鍵暗号回路のアーキテクチャ方式として，各ラウンド処理の間にレジスタを配置し全ラウンド処理をパイプライン処理していくアンロール型アーキテクチャ[6] や，各ラウンドでは同じ処理が繰り返されることから，1 ラウンド分を処理する回路だけ用意しレジスタに中間値を保存しながら繰り返しループさせるループ型アーキテクチャ[7], [8] がある．ここでは，ループ型アーキテクチャ方式を対象としてハードウェア特殊化回路を設計する．本研究では，青木らによって設計されたループ型 Camellia 暗号回路 [9] を評価の基本 (original) として用いることにした．図 3 は，original 回路のブロック図を示している．データレジスタ (Data_reg) への入力をマルチプレクサで切り換え，E$_X$-OR，ラウンド処理回路，FL 関数回路にて順次処理していくことにより暗号化と復号がそれぞれ 23 クロックで実行される．各処理の入力として用いられる副鍵は，鍵拡張部 (Key Scheduler) で生成される．鍵拡張部は，入力暗号鍵を格納するレジスタ KL と，鍵拡張処理によって生成される中間鍵を格納するレジスタ KA，及び各レジスタの出力を 15 bit，または 17 bit 左右にローテンションシフトさせるシフターからなどから構成される．中間鍵を生成する際に，データ処理部と同じくラウンド処理が繰り返し実行されるが，この処理には，暗号化・復号で用いられるラウンド処理回路が共用されている．

4. Camellia 暗号回路のハードウェア特殊化

4.1 副鍵固定回路 (fixed_subkey)

青木らのループ型 Camellia 暗号回路 (original) をもとに，入力暗号鍵を定数に固定したハードウェア特殊化回路を 2 種類提案する．まず，一つ目の回路として，入力暗号鍵が決まれば 26 個の副鍵は事前に計算できることから，全ての副鍵を定数に固定した副鍵固定回路 (fixed_subkey) を提案する（図 4）．original 回路の鍵拡張部に置き換え，副鍵 (kw_1〜kw_4，kl_1〜kl_4，k_1〜k_{18}) を選択するマルチプレクサを配置する．

4.2 F 関数 Ex-OR 簡単化回路 (F_Func_xor_collapse)

排他的論理和処理 (Ex-OR) は，一方の入力が ‘0’ ならばもう一方の入力の値がそのまま出力され，一方の入力が ‘1’ ならばもう一方の入力の値が反転されて出力される．すなわち，片方の入力を定数に固定することによって，Ex-OR 素子を省略または NOT 素子に置き換えることができ，論理回路が簡単化できる．Camellia 暗号では式 (1) で示したとおり，F 関数内

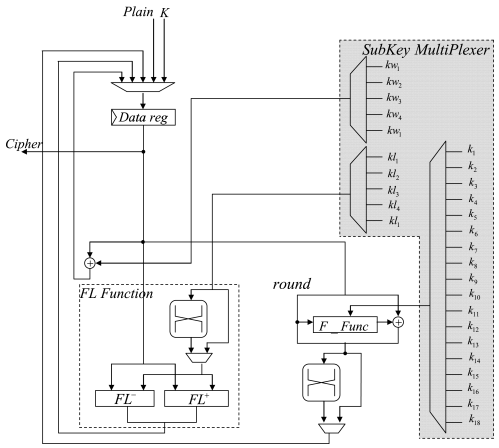

図 4　副鍵固定回路
Fig. 4　fixed_subkey circuit.

で副鍵との排他的論理和処理が行われている．ここでは，全ての副鍵をあらかじめ計算しておき定数に固定することで，F 関数内の排他的論理和 (Ex-OR) を簡単化した回路 (F_Func_xor_collapse) を提案する（図 5）．また，鍵拡張部は副鍵固定回路と同じくマルチプレクサで選択する方式とする．

5. AES 暗号回路のハードウェア特殊化

Camellia 暗号回路との比較のために，ここでは AES 暗号回路についても先行研究をもとにハードウェア特殊化回路を設計した．AES 暗号は SPN 構造を採用しているため，暗号化と復号で異なる回路が必要となるが，ここでは暗号回路だけを対象として回路を設計する．評価の基本 (original) として，Megarajan と Park のループ型 AES 暗号化回路 [10], [11] を用いる（図 6）．全 11 ラウンドの処理が 11 クロックで実行される．original 回路をもとに，Camellia 暗号回路のハードウェア特殊化と同様な方法で，ラウンド鍵を固定した回路 (fixed_roundKey)，ラウンド鍵との排他的論理和処理を簡単化した Ex-OR 簡単化回路 (xor_collapse) を設計した．

6. FPGA による実装評価

前章で述べた AES 及び Camellia のループ型暗号回路とそのハードウェア特殊化回路を Xilinx 社製の FPGA である XC3S400A をターゲットデバイスとして ISE11.1 を用いて論理合成及び配置配線した．論理合成オプションの設定項目の一つである Optimization Goal は Speed 優先と Area 優先の二つの条件についてそれぞれ評価を行い，その他の設定項目はデフォルトのままとする．EDA ツールの動作環境は，CPU が Intel Core2Duo T7250，メモリが 1 GByte，OS が Microsoft Windows XP Professional Service Pack2 の仕様の PC を用いた．

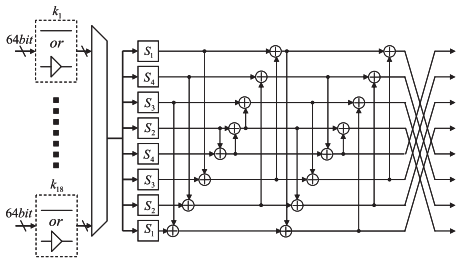

図 5　F 関数簡単化回路（F 関数内部）
Fig. 5　F_Func_xor_collapse circuit in the F function.

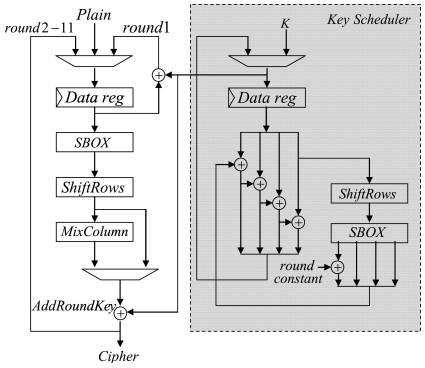

図 6　Megarajan と Park のループ型 AES 暗号化回路
Fig. 6　Megarajan and Park's loop type AES encryption circuit.

鍵を固定したハードウェア特殊化回路は，鍵の定数値ごとに異なる回路が生成される．そこで，ランダムな暗号鍵を 100 種類生成し，それぞれの暗号鍵ごとにハードウェア特殊化を行う．生成された回路ごとに論理合成・配置配線を行い，その平均値を評価結果とする．鍵数は多い方が良いが，今回の実験では 100 個で十分安定した結果が得られた．評価項目は，論理規模 (Logic Scale)，最大動作周波数 (Max.Freq)，スループット (Throughput) 及び，論理合成・配置配線にかかる時間 (Gene.time) とした．ハードウェア特殊化回路では，暗号鍵を入れ換えるたびに再実装する必要があることから，回路生成時間が重要になる．以上の評価環境で AES 暗号回路について論理合成・配置配線した結果を表 1 に示す．Megaration の暗号回路 (original) と比較して，鍵を固定した回路 (fixed_roundKey, xor_collapse) では回路生成時間と

表 1　AES 暗号回路の実装結果
Table 1　Implementation results of AES encryption circuit.

	Design	Gene.Time (sec)	Logic Scale (slices)	Max. Freq. (MHz)	AT Product (slice*msec)	Throughput (Mbps)
speed	original	138	2327	154.9	15.0	1803
	fixed_round_key	121	1683	129.5	12.9	1507
	xor_collapse	120	1742	125.1	13.9	1455
Area	original	129	1828	124.6	14.6	1450
	fixed_round_key	109	1488	120.0	12.4	1396
	xor_collapse	115	1627	109.8	14.8	1278

表 2　Camellia 暗号回路の実装結果
Table 2　Implementation results of Camellia encryption circuit.

	Design	Gene.Time (sec)	Logic Scale (slices)	Max. Freq. (MHz)	AT Product (slice*msec)	Throughput (Mbps)
speed	original	177	1909	71.7	26.6	399
	fixed_sub_key	116	1404	123.0	11.4	685
	F_func xor_collapse	120	1498	147.6	10.1	821
Area	original	141	1751	66.1	26.4	368
	fixed_sub_key	103	1227	104.0	11.7	579
	F_func xor_collapse	119	1441	113.9	12.6	634

論理規模が改善され，動作周波数とスループットが若干低下した．先行研究 [4] と結果の傾向が異なるが，その理由はもとにした回路設計が異なること，特に SubBytes を BlockRAM で実装したか否かの差であると考えられる．本研究のように BlockRAM を用いない場合，Look-up-table で SubBytes を実装するため全体の slice 数が増大し，ハードウェア特殊化による削減効果が（相対的に）小さくなる．

面積時間積（AT 積）は，面積と性能のトレードオフを評価する指標で，小さいほどよい．AT 積を最小化するのは Area 最適化で生成した fixed_round_key 回路で，original 回路の 86%に改善されている．先行研究 [4] ほど劇的ではないが，AES 暗号回路におけるハードウェア特殊化の効果が確認できた．

次に，Camellia 暗号回路の評価結果を表 2 に示す．Speed 最適化でも Area 最適化でも，original 回路と比較して fixed_subkey と F_func_xor_collapse の論理規模は縮小し，動作周波数とスループットが大きく改善されている．論理規模が最小化となるのは fixed_sub_key （Area 最適）で，original に対して論理規模が 30%削減されている．また性能を最大化するのは F_func_xor_collapse （Speed 最適）で，F_func_xor_collapse の動作周波数は original の 2.05 倍に達する．AT 積を最小化するのも F_func_xor_collapse （Speed 最適）で，original の 38%（62%削減）となっている．fixed_subkey の方が xor_collapse よりも回路規模に関しては改善効果が大きかった．また，AES 暗号と比較すると，Camellia 暗号におけるハードウェア特殊化の効果は非常に大きい．これらのハードウェア特殊化回路による性能改善効果の相違については，今後詳細に原因を調査する予定である．

7. む す び

本研究では，Camellia 暗号回路について，暗号鍵を定数に固定したハードウェア特殊化回路を提案した．その結果 Xilinx FPGA で論理合成・配置配線した結果，ハードウェア特殊化する前の回路と比較して，回路生成時間，論理規模，最大周波数，スループットともに改善されることが確認できた．

文　献

[1] C. Consel and O. Danvy, “Tutorial notes on partial evaluation,” Proc. 20th ACM Symposium on Principles of Programming Language, pp.493–501, 1993.

[2] N.D. Jones, “An introduction to partial evaluation,” ACM Computing Surveys, vol.28, no.3, pp.480–503, 1996.

[3] J. Leonard and W.H. Mangione-Smith, “A case study of partially evaluated hardware circuits: Key-specific DES,” Proc. FPL’97, LNCS 1304, pp.151–160, Springer, 1997.

[4] R. Atono and S. Ichikawa, “Design and evaluation of data-dependent hardware for AES encryption algorithm,” IEICE Trans. Inf. & Syst., vol.E89-D, no.7, pp.2301–2305, July 2006.

[5] K. Aoki, T. Ichikawa, M. Kanda, M. Matsui, S. Moriai, J. Nakajima, and T. Tokita, “Specification of Camellia a 128 bit block cipher,”

https://info.isl.ntt.co.jp/crypt/camellia/specifications.html/01jspec.pdf, 2000.

[6] D. Denning, J. Irvine, and M. Devlin, "A high throughput FPGA Camellia implementation," PhD Research In Micro-Electronics & Electronics, pp.25–28, 2005.

[7] H. Cheng and H. Heys, "Compact hardware implementation of the block cipher Camellia with concurrent error detection," Canadian Conference on Electrical and Computer Engineering, pp.1129–1132, 2007.

[8] P. Yalla and J.P. Kaps, "Compact FPGA implementation of Camellia," Proc. FPL 2009, pp.658–661, 2009.

[9] 東北大学青木研究室, "Camellia IP core," http://www.aoki.ecei.tohoku.ac.jp/crypto/web/cores.html, 2010.

[10] Megarajan and Park, http://islab.oregonstate.edu/koc/ece575/03Project/, 2009.

[11] M.B. Abdelhalim and H.K. Aslan, A Design for an FPGA Implementation of Rijndael Cipher, ICGST-PDCS, vol.9, pp.9–15, 2009.

（平成 23 年 2 月 23 日受付，6 月 7 日再受付）