ストリーム暗号 Enocoro-128v2 のソフトウェアおよびハードウェア実装と評価

三上　修吾　　渡辺　大

株式会社　日立製作所　横浜研究所
244-0817 神奈川県横浜市戸塚区吉田町 292 番地
shugo.mikami.hj@hitachi.com

あらまし　本稿ではストリーム暗号 Enocoro-128v2 のソフトウェア及びハードウェア実装性能を示す．Enocoro-128v2 は株式会社日立製作所が開発した軽量ストリーム暗号で，CRYPTREC 電子政府推奨暗号リストへ提案されている．ソフトウェア実装では，32 ビット CPU の Intel Core2 Duo を対象 CPU とし，C 言語またはアセンブリ言語により高速実装を行い，処理速度は 14.8 cycle/byte であった．ハードウェア実装では軽量実装と ASIC による実装評価を行い，回路規模は 2.4 kGE であった．

Software and Hardware Implementation and Evaluation of Stream Cipher Enocoro-128v2

Shugo Mikami　　Dai Watanabe

††Hitachi, Ltd., Yokohama Research Laboratory
292, Yoshida-cho, Totsuka-ku, Yokohama-shi, Kanagawa-ken, 244-0817 JAPAN
shugo.mikami.hj@hitachi.com

Abstract　In this paper, we describe software and hardware implementations and evaluation results of Enocoro-128v2, which is a lightweight stream cipher proposed by Hitachi, Ltd., and is submitted to Cryptography Research and Evaluation Committees (CRYPTREC). We estimate Enocoro-128v2 on an 32-bit CPU Intel Core2 Duo in C and assembly language and it is shown that the speed of Enocoro-128v2 achieves 14.8 cycle/byte. We also estimate gate count of Enocoro-128v2 on ASIC and show that it requires 2.4 kGE.

1　はじめに

組み込み機器においても個人情報や課金情報などの秘密情報がやり取りされるようになり，これらの秘密情報を盗聴や改ざんといった脅威から保護するためにも，組み込み機器へ適用可能な暗号技術が求められている．組み込み機器は携帯電話などのように大量の情報をリアルタイムに処理する必要のある分野や，スマートグリッドで使用するセンサなどのように軽量実装が必要とされる分野での使用が見込まれる．従って，組み込み機器に適用する暗号プリミティブにも高速処理性や軽量実装性が求められる

Enocoro-128v2[1] [3] はソフトウェアでの高速処理性とハードウェアでの軽量実装性の両立を目指して，2010 年に日立製作所が開発したストリーム暗号である．Enocoro-128v2 は CRYP-

1. Enocoro は株式会社日立製作所の登録商標です．

TREC による電子政府推奨暗号リスト改訂のための暗号技術の公募へ提案され，安全性や実装性能が評価されている．

Enocoro-128v2 の安全性評価に関しては多くの報告があるが [3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]，実装評価に関しては我々の知る範囲では自己評価書 [3] に記載の評価結果のみである．自己評価書に記載のソフトウェア実装では C 言語による評価結果が示されているが，アセンブリ言語の利用は行っていない．また，ハードウェア実装では不要なハードウェア素子の省略などは行っていないと考えられる．以上より，Enocoro-128v2 のソフトウェアおよびハードウェア実装性能は更に向上する余地があると考えられる．

本稿では Enocoro-128v2 のソフトウェアおよびハードウェア実装性能の評価を行う．ソフトウェア実装では 32 ビット CPU の Intel[1]Core2 Duo[2]を対象 CPU とし，アセンブリ言語または C 言語により速度最適化を目指した実装を行い，実装性能を評価する．ハードウェア実装では不要なハードウェア素子の省略により規模最適化を目指した実装を行い，ASIC 実装性能を評価する．そして，今回の実装に基づき，Enocoro-128v2 と AES の実装性能を比較する．

今回の実装により，ソフトウェア実装における処理速度は 14.8 cycle/byte となり，従来の評価結果より 3.1 倍の高速化を達成した．また，ハードウェア実装における論理規模は 2.4 kGE となり，従来の評価結果より 42 % の軽量化を達成した．AES との比較では，特にハードウェア実装における論理規模は同等であった．

本稿では Enocoro-128v2 を Enocoro と略記する．以下，2 節にて Enocoro の仕様を記載する．3 節にて Enocoro のソフトウェア実装とその評価結果を示す．4 節にて Enocoro のハードウェア実装その評価結果を示す．5 節にて Enocoro と AES の実装性能を比較し，6 節でまとめを行う．

2 Enocoro の仕様

本節では，Enocoro の仕様を記載する．Enocoro は株式会社日立製作所が 2010 年に開発したストリーム暗号である．Enocoro の入力は，秘密鍵 128 ビットと初期ベクトル (以下，IV と記す)64 ビットである．Enocoro は 272 ビットの内部状態を持ち，更新関数を用いて内部状態を更新しつつ，96 ラウンドの初期化を行った後，8 ビットずつ擬似乱数を出力する．詳細な仕様については，Enocoro 仕様書 [3] を参照のこと．

2.1 記号とデータ構造

Enocoro の仕様を記述するために必要な記号とデータ構造をまとめて示す．

|| : 2 つのビット列の連結

<<< n : 左回りで n ビット巡回シフト

$\oplus$: 排他的論理和

GF(2^n) : 有限体

また，Enocoro では 1 バイトを基本単位とする．

2.2 Enocoro の仕様

2.2.1 内部状態

Enocoro はステートとバッファの 2 つの内部状態を持つ．ステート a は 2 バイトからなり，上位から a_0, a_1 と表す．バッファ b は 32 バイトからなり，上位から $b_0, b_1, \cdots, b_{31}$ と表す．本稿では，時刻 t における i バイト目のステートを a_i^t，j バイト目のバッファを b_j^t と書く．

2.2.2 ρ 関数

内部状態の更新関数として，ステートの更新関数 ρ とバッファの更新関数 λ を備える．ステートの更新関数である ρ 関数は，8 ビット入出力の非線形変換 Sbox s_8，線形変換 L と XOR か

1. Intel は Intel Corporation の登録商標です.
2. Core2 は Intel Corporation の商品名称です.

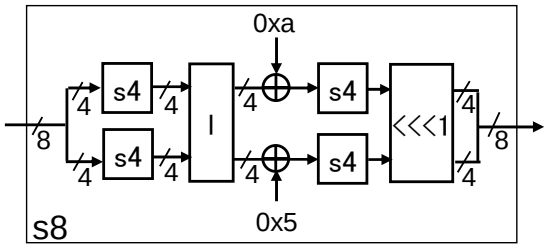

図 1: Sbox s_8 の構成

ら構成され，以下のように記述される．

$$
\begin{aligned}
u_0 &= a_0^t \oplus s_8[b_2^t],\\
u_1 &= a_1^t \oplus s_8[b_7^t],\\
(v_0, v_1) &= L(u_0, u_1),\\
a_0^{t+1} &= v_0 \oplus s_8[b_{16}^t],\\
a_1^{t+1} &= v_1 \oplus s_8[b_{29}^t].
\end{aligned}
$$

ここで，u_0, u_1, v_0, v_1 は一時的な変数である．

(1)Sbox s_8

Sbox s_8 は 4 ビット入出力の非線形変換 s_4，線形変換 l，定数 XOR と左回り 1 ビット巡回シフトから構成される．s_4 は以下で定義される．

$$
\begin{aligned}
s_4[16] = \{&1, 3, 9, 10, 5, 14, 7, 2,\\
&13, 0, 12, 15, 4, 8, 6, 11\}.
\end{aligned}
$$

l は GF(2^4) 上の行列を用いて，以下の式で定義される．

$$
l(x, y) = \begin{pmatrix} 1 & 4 \\ 4 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}.
$$

これらの構成要素を用いて，Sbox s_8 は図 1 のように構成される．また，式で記述すると次のようになる．

$$
\begin{aligned}
y_0 &= s_4[s_4[x_0] \oplus 4 \cdot s_4[x_1] \oplus 0\mathrm{x}a],\\
y_1 &= s_4[4 \cdot s_4[x_0] \oplus s_4[x_1] \oplus 0\mathrm{x}5],\\
s_8[x] &= (y_0 || y_1) <<< 1.
\end{aligned}
$$

ここで，x_0 は入力 x の上位 4 ビットを表し，x_1 は入力 x の下位 4 ビットを表す．同様に，y_0 は出力 y の上位 4 ビットを表し，y_1 は出力 y の下位 4 ビットを表す．

(2) 線形変換 L

線形変換 L は GF(2^8) 上の行列を用いて，次の式で定義される．

$$
(v_0, v_1) = L(u_0, u_1) = \begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix} \begin{pmatrix} u_0 \\ u_1 \end{pmatrix}.
$$

2.2.3 λ 関数

バッファの更新関数である λ 関数は次の式で定義される．

$$
\begin{aligned}
b_i^{t+1} &= b_{i-1}^t, \qquad i \neq 0, 3, 8, 17,\\
b_0^{t+1} &= b_{31}^t \oplus a_0^t,\\
b_3^{t+1} &= b_2^t \oplus b_6^t,\\
b_8^{t+1} &= b_7^t \oplus b_{15}^t,\\
b_{17}^{t+1} &= b_{16}^t \oplus b_{28}^t.
\end{aligned}
$$

2.2.4 出力

ステート a_1 を出力する．

2.2.5 初期化

Enocoro の初期化は，128 ビットの秘密鍵，64 ビットの IV と 80 ビットの定数を，以下のように内部状態へ入力する．

$$
\begin{aligned}
b_i &= K_i, \qquad 0 \leq i < 16,\\
b_{i+16} &= I_i, \qquad 0 \leq i < 8,\\
b_{i+24} &= C_i, \qquad 0 \leq i < 8,\\
a_0 &= C_8, \quad a_1 = C_9.
\end{aligned}
$$

ここで，K_i は 1 バイト長に分けた秘密鍵である．同様に I_i は 1 バイト長に分けた IV，C_i は 1 バイト長に分けた定数である．

その後，カウンタ値の XOR と更新関数の適用を 96 回繰り返す．カウンタ値は b_{31} に XOR される．尚，カウンタ値は初期値が 1 で，GF(2^8) 上の 2 倍算で更新される．

初期化時の Enocoro アルゴリズム図を図 2 に示す．図中，S は Sbox s_8，L は線形変換 L を

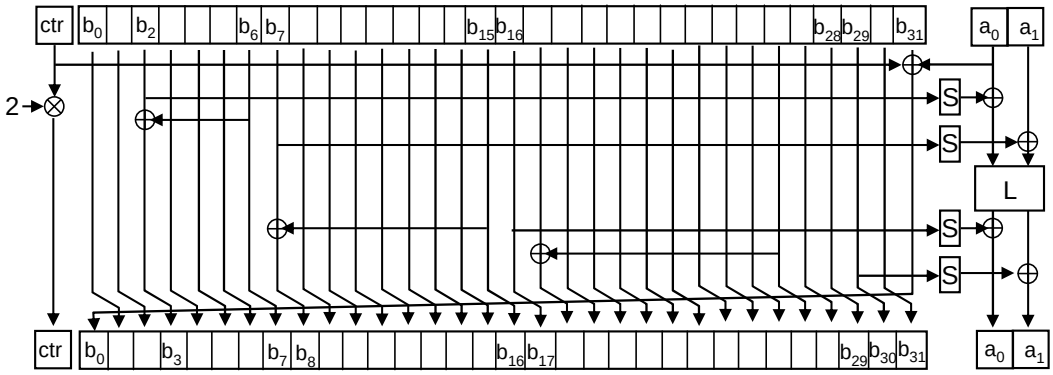

図 2: Enocoro 初期化におけるアルゴリズム図

示す．また，上段の ctr, a_i, b_j はそれぞれ更新前のカウンタ，ステート，バッファを示し，それぞれの値は結線に従って更新される．下段の ctr, a_i, b_j はそれぞれ更新後のカウンタ，ステート，バッファを示す．

3 Enocoro のソフトウェア実装

Enocoro のソフトウェア実装を 32 ビットプロセッサである Intel Core2 Duo を対象に行った．言語は C 言語またはアセンブリ言語を用いた．

3.1 C 言語による実装

3.1.1 初期化関数と乱数生成関数

C 言語を用いて，初期化関数と乱数生成関数を実装した．初期化関数，乱数生成関数は 32 段展開とした．32 バイトで割り切れない端数の処理については，32 段展開の乱数生成関数を利用して 32 バイトの乱数を生成したのち，トランケート処理を行い出力する．

さらにバッファの右巡回シフト処理を先頭インデックスで代用する高速化を施している．

3.1.2 ρ 関数の実装

Enocoro の ρ 関数で使用する Sbox 変換と線形変換 L の 2 倍算については，8 ビット入力 8 ビット出力を行う 256 バイトの表参照を利用して実装した．Sbox 変換と 2 倍算それぞれにおいて参照表を 1 枚ずつ用いるため，参照表のサイズは合計で 512 バイトとなる．

3.2 アセンブリ言語による実装

3.2.1 初期化関数と乱数生成関数

1 段分の初期化関数と乱数生成関数をアセンブリ言語により実装した．96 段展開を行う初期化関数ではアセンブリ言語で実装した関数を 96 回実行する．同様に 32 段展開を行う乱数生成関数ではアセンブリ言語で実装した関数を 32 回実行する．

3.2.2 ρ 関数の実装

Enocoro の ρ 関数で使用する Sbox 変換と線形変換 L の 2 倍算については，8 ビット入力 8 ビット出力を行う 256 バイトの表参照を利用して実装した．Sbox 変換と 2 倍算それぞれにおいて参照表を 1 枚ずつ用いるため，参照表のサイズは合計で 512 バイトとなる．

3.3 実装結果

評価環境を表 1 に示す．速度測定は eBASC[2] プロジェクトにて公開されている評価ツール

表 3: Core2 Duo 上での実装性能

データサイズ (bytes)	0(初期化)	8	64	576	1536	4096	long message
C 言語 (cycle/byte)	1944 cycles	291.4	48.4	21.4	19.9	19.3	19.0
アセンブリ言語 (cycle/byte)	1530 cycles	248.6	39.5	18.3	16.0	15.3	14.8

表 1: 評価環境

CPU	Intel Core2 Duo E6600 (2.4GHz)
メモリ	4GB
OS	Ubuntu Linux 8.04 32-bit
コンパイラ	gcc 4.2.4

SUPERCOP (version 2011. 05. 22) を利用した．C 言語，アセンブリ言語によるメモリ使用量を表 2 に示し，処理速度を表 3 に示す．

表 2: メモリ使用量

	ROM (bytes)	RAM (bytes)
C 言語	11523	110
アセンブリ言語	14846	178

long messages の値のみ 1536 バイトと 4096 バイトの処理サイクル数の差分を 2560 で除算することで求めている．

自己評価書 [3] では 1Mbyte のデータを処理した際の処理速度は 46.3 cycle/byte であった．従って，今回の実装により処理速度が約 3.1 倍向上している．

4 Enocoro のハードウェア実装

本節では，Enocoro のハードウェア実装および実装性能評価について述べる．

4.1 実装アーキテクチャ

Enocoro の実装アーキテクチャは，アルゴリズム通りの構成，即ちステートとバッファをそれぞれ 2 バイト，32 バイトのレジスタとして実装する．また λ 関数と ρ 関数を 1 個ずつ組み合わせ論理として実装し，1 クロックで 1 サイクル分のラウンド処理を行う．

また，ハードウェアの使用方法が制限されるものの，レジスタが自身の値を保持するためのフィードバック回路を省略することで軽量化を行う．

4.1.1 Sbox の実装

Enocoro の Sbox は図 1 に示すように，4 個の 4 ビット Sbox s_4 と線形変換 L，定数との XOR と左 1 ビット巡回シフトの組合せで構成する．この構成とすることで，8 ビット入出力の表参照として実装するよりも信号遅延時間は長くなるものの小論理規模での実装が可能となる．今回は軽量実装を目的としているため，Sbox は組み合わせ論理回路で実装する．

4.1.2 初期値の入力方法

初期値の入力方法は，すべてのバッファに同時に秘密鍵や IV を入力する方法と，初期値を一定長ずつに分割し逐次的に入力し他のレジスタへシフトする方法が考えられる．前者はすべてのレジスタに初期値の入力線とセレクタが必要である．後者は初期値の入力線を備えるレジスタは一定長分のみでよいが，シフトレジスタ構造が求められる．

Enocoro のバッファは 1 バイト単位の右巡回シフトを基本構成としているため，シフトレジスタ構造を備えている．そのため，秘密鍵と IV を 1 バイトずつ逐次的に入力するために追加で必要なハードウェア素子は少なくて済む．従って初期値の入力方法は，秘密鍵と IV を 1 バイトずつ左端バッファ b_0 へ入力し，シフトレジス

タにより他のバッファレジスタ b_i へシフトさせることにする．

4.2 実装結果

上記の方針に従って Enocoro を verilog で記述し，Application Specific Integrated Circuit (ASIC) 向けに論理合成し，論理規模とスループットの評価を行った．Synopsys[1]Design Compiler [2] (Version C-2009.06-SP5) で論理合成を行った．ライブラリには TSMC 社の 90 nm high performance ライブラリを使用した．また，入力データの遅延時間を 0.4 ns，出力データの遅延時間を 0.4 ns に設定した．表 4 に ASIC 実装評価結果を示す．

自己評価書 [3] では，規模優先での論理規模は 4.1 kGE，スループットは 3500 Mbps であった．従って，今回の実装により論理規模を 42% 削減でき，またスループットも 1.78 倍に向上した．

5 AES との比較

本節では Enocoro と AES[14] のソフトウェア実装性能及びハードウェア実装性能を比較する．K*ä*sper らによると AES の Intel Core 2 Quad Q6600 (2.4GHz) 上での処理速度は 9.32 cycle/byte であった [15]．CPU コア数の違いがあるため厳密な比較は困難であるが，Enocoro は AES より約 1.6 倍遅い結果となった．

また，Moradi らによると AES の 130 nm プロセス幅のハードウェアによる軽量実装では回路規模は 2.4 kGE であった [16]．こちらもプロセス幅の違いがあるため厳密な比較は困難であるが，Enocoro と AES の回路規模はほぼ同等となった．

6 まとめ

本稿ではストリーム暗号 Enocoro-128v2 のソフトウェア実装およびハードウェア実装を行い，その性能を評価した．ソフトウェア実装では C 言語とアセンブリ言語を用いた高速実装を行い，32 ビットプロセッサの Intel Core2 Duo 上での処理速度は 14.8 cycle/byte であった．従来の評価結果より 3.1 倍の高速化を達成した．ハードウェア実装では軽量実装を行い，90 nm CMOS スタンダードセルライブラリを用いた ASIC 実装評価では論理規模は 2.4 kGE であった．従来の評価結果より 42 % の軽量化を達成した．AES のソフトウェア実装及びハードウェア実装結果と比較すると，特にハードウェア実装では同等の論理規模を達成することを確認した．

参考文献

[1] The eSTREAM Project, available at `http://www.ecrypt.eu.org/stream/`.

[2] ECRYPT Benchmarking of Cryptographic Systems, available at `htp://bench.cr.yp.to/`

[3] Hitachi,Ltd. Stream Cipher Enocoro, available at `http://www.hitachi.com/rd/yrl/crypto/enocoro/index.html`.

[4] 渡辺大，岡本和人，金子敏信，“軽量ハードウェア向け疑似乱数生成器 Enocoro-128v2”，SCIS2010，3D1-3，2010.

[5] 岡本和人，武藤健一郎，金子敏信，“疑似乱数生成器 Enocoro-80 の差分／線形攻撃耐性評価（II）”，SCIS2009，4B2-3，2009.

[6] 武藤健一郎，渡辺大，金子敏信，“Enocoro-128 の LDA 耐性評価と改良”，SCIS2008，4A1-1，2008.

[7] 武藤健一郎，渡辺大，金子敏信，“Enocoro-80 の再同期攻撃（線形攻撃）耐性評価”，SCIS2008，4A1-2，2008.

[8] 古市洋希，武藤健一郎，渡辺大，金子敏信，“擬似乱数生成器 Enocoro-80 の再同期攻撃（差分攻撃）に対する耐性評価”，SCIS2008，4A1-3，2008.

1. Synopsys は Synopsys 社の登録商標です．
2. Design Compiler は Sysnopsys 社の商品名称です．

表 4: ASIC 実装評価結果

コンパイル条件	論理規模 (kGE)	動作周波数 (MHz)	スループット (Mbps)
規模優先	2.40	781.3	6250.0
速度優先	3.17	1408.5	11267.6

[9] 井手口恒太, 渡辺大, “推測決定攻撃に対する安全性評価の一手法”, SCIS2008, 3A1-4, 2008.

[10] D. Watanabe, K. Ideguchi, J. Kitahara, K. Muto, H. Furuichi, T. Kaneko, “Enocoro-80: A Hardware Oriented Stream Cipher,” Second International Workshop on Advances in Information Security, 2008.

[11] 鴻巣慧, 武藤健一郎, 古市洋希, 渡辺大, 金子敏信, “Enocoro-128 ver.1.1 の再同期攻撃耐性評価”, ISEC2007-147, 2008.

[12] 渡辺大, 金子敏信, “軽量な PANAMA 型疑似乱数生成器の構成に関する検討”, ISEC2007-78, 2007.

[13] 武藤健一郎, 渡辺大, 金子敏信, “疑似乱数生成器 Enocoro-80 の Linear Distinguish Attack 耐性評価”, SITA2007 2.4, 2007.

[14] NIST, Federal Information Processing Standards Publication 197, ADVANCED ENCRYPTION STANDARD (AES), 2001.

[15] E. Käsper, and P. Schwabe, “Faster and Timing-Attack Resistant AES-GCM”, CHES 2009, Lecture Notes in Computer Science, vol.5747, pp.1-17, Springer-Verlag, 2009.

[16] A. Moradi, A. Poschmann, S. Ling, C. Paar, and H.Wang, “Pushing the Limits: A Very Compact and a Threshold Implementation of AES”, Eurocrypt 2011, Lecture Notes in Computer Science, vol.6632, pp.69-88, Springer-Verlag, 2011.