PPRM ベース形式手法を用いた AES 内演算回路の最適化検証

森岡 澄夫

NEC 中央研究所
211-8666 神奈川県川崎市中原区下沼部 1753
s-morioka@ak.jp.nec.com

あらまし AES 暗号回路の実装に誤りがない事を 100%保証したい．個別演算器の検証に絞っても，MixColumns，マスキングなどサイドチャネル攻撃対策を施した演算器，用いる基底を変えた演算器等では，総当りが依然難しかったり正しい仕様（リファレンス）をどのように作るかが問題となったりする．従来，有限体が多用される暗号回路向けの形式検証手法はほとんどなかったが，近年，本間らが Gröbner 基底を用いた手法を提案するなど，報告が増え始めた．本稿では，リファレンスをボトムアップ構築しつつ，論理式の PPRM(Positive Polarity Reed-Muller) 変換を用いて，演算器が正しく最適化されている事を短時間で検証した事例を示す．

Verification of AES component H/W optimization using a PPRM-based formal method

Sumio Morioka†

Central Research Laboratories, NEC Corporation
1753 Shimonumabe, Nakahara-ku, Kawasaki, Kanagawa 211-8666, JAPAN
s-morioka@ak.jp.nec.com

Abstract In this paper, we propose a formal verification (FV) method which is suitable for AES H/W verification. Exhaustive testing based on dynamic simulation, even for partial components such as MixColumns, side-channel resistant masked operators and some other optimized operators, is difficult. Another important issue is how to construct correct specifications, or reference, of these components. In this paper, we demonstrate (i) bottom up construction of the reference and (ii) fast correctness proof of optimized AES components. A simple decision procedure which convert logic formula into PPRM (Positive Polarity Reed-Muller) form is used.

1 はじめに

1.1 AES 実装検証の重要性の高まり

AES 暗号 [1] は，クラウド・コンピューティングやセンサネットワーク等の発展に伴い，今後も利用が拡大していくと期待される．ソフトや回路（ASIC や FPGA）による実装に関しても，速度や消費リソースの最適化法が多数研究されている．

以下，おもに ASIC での回路実装を想定する．ASIC 実装は，組み込み機器や IC カード等の低電力デバイスや，耐タンパ性の要求される装置に広く用いられる．ASIC は製造・市場投入後の修正が不可能なため，フロントエンド設計の段階でバグを確実に排除しなければならない．

それにも関わらず，従来，AES 回路の設計検

証法[1] について，活発な研究が行われてきたとは言い難い．おもな理由として:
(1) コーナーケースが存在しないと考えられ，実際に，シミュレーションで数個のテストケースが通過すればまずバグが出ない，という経験則もある，
(2) どれだけのテストをすれば十分か基準が不明確（コード・カバレッジやゲート・トグル率を基準とするのが適切かは疑わしい），
(3) 有限体演算を扱うこともあり，検証分野ではニッチな対象と捉えられてきた，
等が考えられる．

しかし，上記経験則に明確な根拠があるわけではない事や，AES 回路の開発機会が拡大している事，サイドチャネル攻撃対策などで AES 実装形態が多様化している事などから，確実な設計検証を行う方法を確立したい．

1.2 関連研究

本稿では形式検証の適用を検討する．これは，回路実装が何らかの性質を満たすこと（今回は，AES 仕様どおりの演算を行うこと）を形式論理を用いて判定する手法である．シミュレーションやテストと異なり，全入力空間の総当り検証をすることなく，当該性質の成立を 100%保証できる点が特徴である．一般の算術演算回路の形式検証については古くから多くの研究があり，実用 ASIC の設計にも広く用いられている[2].

有限体上の演算回路の形式検証は 10 年ほど前から報告が出始めた．筆者らは文献 [4, 5] において，$GF(2^m)$（拡大体も含む）上の一階述語論理式のあるサブクラスを定義し，ルール・ベースの定理証明と FDD[3] とを併用した真偽決定手法を提案し，誤り訂正回路の自動検証を行った．また，文献 [6, 7, 8, 9, 10] では，合成体 $GF((2^m)^n)$ における上記手法の効率化や，乗算回路の形式検証などが試みられている．

AES 暗号回路の形式検証の試みがなされるようになったのは，さらに最近である．文献 [11, 12, 13] では，CPU 命令拡張やソフトウェアによる実装の検証が行われている．注目すべき事例として，文献 [14, 15, 16] において本間らが AES の仕様・実装記述法を定め，Gröbner 基底を用いた自動検証手続きにより短時間（10 分強）で検証を行えることを示した．

ただ，任意の設計を常に高速に判定できる検証手段はない事が，経験的に知られている．とくに形式検証においては，設計が正しい場合は非常に高速でも，設計誤りがあると処理時間が指数的に増加する場合がしばしばある．これは実用上は大きな支障となる．

また，もう一つの重要な問題として，検証に用いる「正しい仕様」をいかに作るかが挙げられる（シミュレーションにおいても，正しいテストデータを作る方法が問題だが，それと同様）．

1.3 本研究の概要

そこで本稿では，PPRM 形式（Positive Porality Reed Muller Form）の命題論理式の等価性判定を用いて，AES 回路部品の最適化の正しさを検証した事例を示す．PPRM は AND と XOR を用いた論理式表現の一種である [3, 4].

本手法の特徴として，(A) 正しい仕様をボトムアップで構築しつつ検証を行う，(B) 基底や体の変換，サイドチャネル攻撃対策などを施した有限体演算 [17, 18, 19, 20, 22, 23] であっても，仕様と実装の両方を自然な形で扱える，(C) カノニカル・フォーム[3] であるため，完全自動で検証が行えるとともに，正しい回路を誤りとするような誤判定も起きない，等がある．

検証にかかる時間は，設計誤りがなければ本間らの手法 [14, 15, 16] と同程度である．しかし設計誤りがある場合，判定時間が大きく変わり得るので，たとえば両手法を併用（並列実行）した判定高速化等が好ましい．

以下，**2.** では検証の基本方針について，**3.** では提案する自働形式検証手法について，**4.** では AES 回路の検証結果について述べる．

[1] AES の暗号強度や耐タンパ性の検証ではなく，回路実装が AES 仕様どおりであるかの検証であることに注意.

[2] たとえば，論理合成前の RTL と合成後のネットリストの等価性証明は，ASIC の標準設計フローとして確立している.

[3] 論理式の形式のうち，真理値表が同じなら論理式表現が一意に定まるもの.

2　本手法の狙いと検証項目

2.1　検証の基本アプローチ

形式検証には，図 1 に示すとおり，(A) 仕様の入出力関係と実装の入出力関係をそれぞれ形式論理で記述し，後者が前者を包含する（または等価である）ことを判定する場合と，(B) 実装の入出力関係のもとで，何らかの性質（プロパティ）が成立する事を判定する場合とがある．

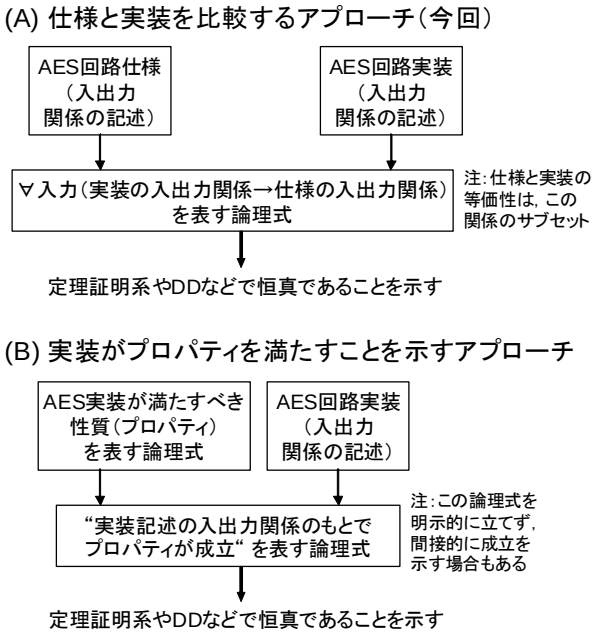

図 1: 形式検証の基本アプローチ

いずれの場合も，「実装が正しい」ことを現実に示せるかにつき，慎重な検討を要する．前者 (A) では，正しい仕様記述を容易に作れるかが問題になりやすい．仕様が実装とほとんど変わらない抽象度・粒度・分量になってしまう場合がしばしばあり，それでは検証する意義が実質的にないからである．後者 (B) では，どのようなプロパティを満たせば「実装が正しい」と結論づけられるのかが問題となる．

2.2　提案手法で実施する検証の意義

本手法では (A) のアプローチを採る．ただし上記のとおり，AES の仕様 [1] を形式論理で記述すると，そのまま一つの AES 実装になってしまう[4]．つまり，教科書的な実装を検証対象とするならば，(検証の作業は可能だが）実質的な意義はあまりない．

しかし，実用に使われる AES 回路では，様々な最適化が行われる．たとえば 1 ラウンド/クロック構成の範疇でも，回路規模やクリティカル・パス遅延を改善するために，以下のアレンジが行われている[5]（一部を図 2 にも示す）：

・ 合成体 $GF(((2^2)^2)^2)$ や $GF((2^4)^2)$ [18]，あるいは多項式基底とは異なる基底を用いる [20].
・ MixColumns も合成体の上で演算し，同型写像は AES の全体入出力に置く．
・ SubBytes のアフィン変換や MixColumns はいずれも XOR マトリックスなので，マージする．
・ または，SubBytes 全体と MixColumns をマージし，Tbox とする [19].
・ MixColumns や InvMixColumns の出力ビット間で XOR 演算を共有する．
・ DPA などのサイドチャネル攻撃対策のため，SubBytes 演算にランダムマスクをかける [22, 23]．MixColumns にもマスク通過処理を施す場合がある．

そこで本稿では，仕様どおりのシンプルな AES（以下，**リファレンス**と呼ぶ）ないしその部品と最適化実装とを比較して，「最適化の正しさを証明する」ことを狙いとする[6]．

3　提案する PPRM ベース形式検証手法

3.1　階層的な検証

本稿では，SubBytes, MixColumns など個別演算器の検証と，それらの接続・制御の検証を

[4] SubBytes，ShiftRows，MixColumns，AddRoundKey をそれぞれ関数として定義し，それらが順次つながることを記述すると，ほぼそのまま 1 ラウンド/クロック構成の AES 実装になってしまう．

[5] もちろん，ループ・アンロール（1 クロックあたりのラウンド数の増加），パイプライン化，シリアライズなど，回路での処理展開上の工夫もなされる．

[6] これは例えば，整数上の高速加算器や高速乗算器を作ったとき，シンプルなリップルキャリー・アダーや配列型乗算器と比較して検証する事に相当する．

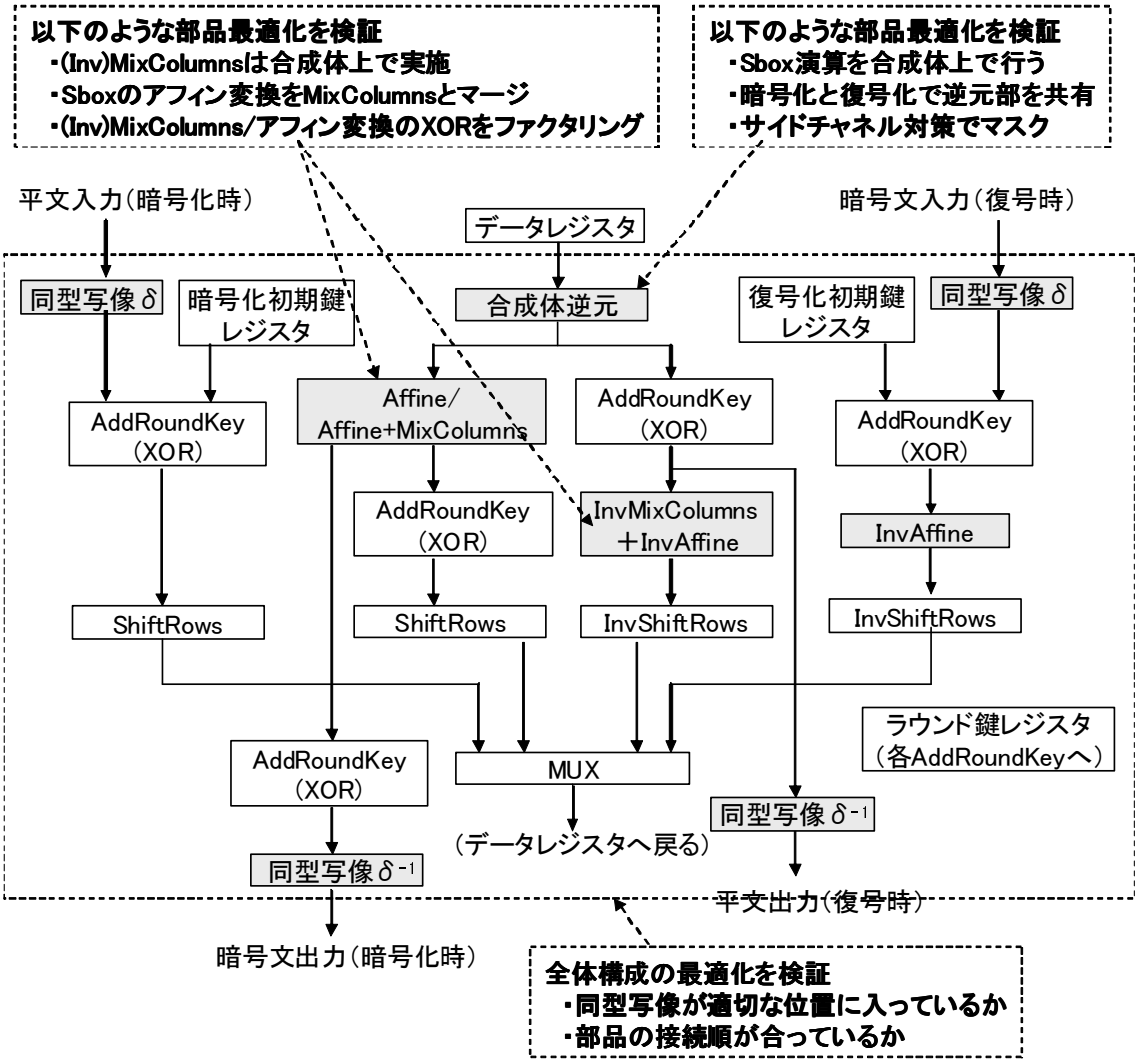

図 2: 実用 AES 回路で行われる最適化と，検証が必要となる箇所（色付き部）

分離して考える．それぞれ，一般の演算回路検証におけるビットレベル検証とワードレベル検証に相当する．

ワードレベル検証の方法：実装記述上の記号実行等により，1 ラウンド分の演算系列 （演算の実行順）を求め，それが AES 仕様と一致するかをワードレベル（$GF(2^8)$ 上）で調べる．文献 [4] の手法により，一般の順序回路検証と同様に行えるので，詳細は割愛する．

なお，AES に特有な重要検証項目としては，
(i) 最終ラウンドで MixColumns をスキップするように，全体制御がなされていること，
(ii) 合成体や基底変換等を使っている場合，同型写像 δ と δ^{-1} が過不足なく対になって出現し，かつ，各演算器が動作する体・基底と実際のデータの体・基底が合致すること，
が挙げられる．

ビットレベル検証の方法：個別演算器を AND, XOR ゲートを組み合わせて実装したとき，任意の入力について AES 仕様と同じ出力が得られることを，ビットレベル（$GF(2)$ 上）で証明する．ただし体や基底の変換を行った演算器では，その入力値と出力値を AES field[7]に写像したうえで調べる．詳細は次節で述べる．

3.2 仕様となるリファレンス演算器の構成

ビットレベル検証では，図 3 に示したとおり，リファレンス演算器を $GF(2)$ 上で構成したう

[7]既約多項式が $x^8 + x^4 + x^3 + x + 1$，多項式基底の $GF(2^8)$.

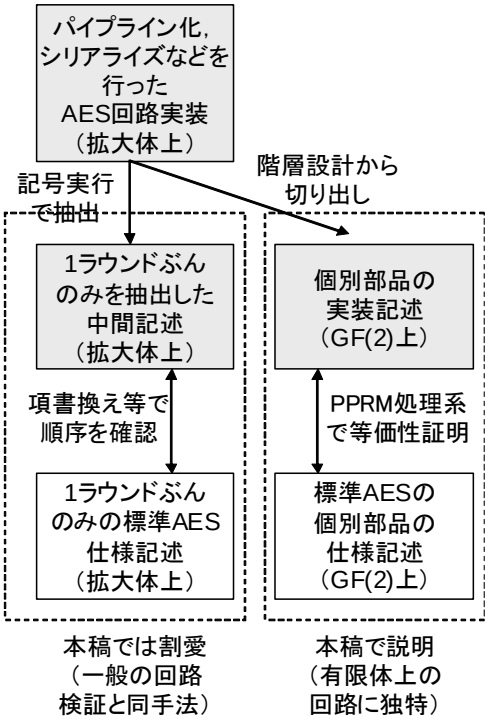

図 3: 今回用いる AES 実装検証の手順

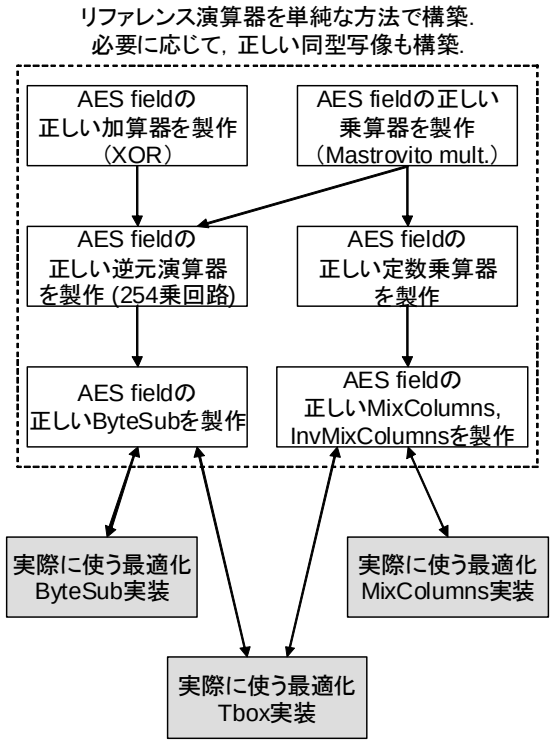

図 4: リファレンス演算器の構成手順

えで実装回路との等価性を判定する．

リファレンス演算器の構成手順を図 4 に示す．最初に，AES field 上の加算器をビットパラレル XOR で，乗算器を単純な Mastrovito Multiplier [21] として作る．いずれも至って規則的な構造を持つので，正しさは記述上で容易に目視確認できる．次に，これらを用いて正しい定数乗算器（乗算器入力に定数値を入れる）や乗法逆元演算器（複数の乗算器で 254 乗する）を構成し，さらに正しい SubBytes や (Inv)MixColumns を構成する．Tbox[19] など複合演算器のモデルも作成できる．

ここまでのポイントは，単純で目視チェック用意な方法のみを用い，最適化を一切施さずにリファレンス演算器を構成することである．

3.3 検証条件を表す論理式の構成

リファレンス演算器と実装演算器との比較を行う．一般に，後者の入出力関係が前者の入出力関係に包含されていれば（等価な場合も含む），実装演算器は正しいと判断できる．

このことは次のような論理式（**検証条件**）で表現できる．演算器入力のビット列 $< x_1, \cdots, x_n >$ と出力ビット列 $< y_1, \cdots, y_m >$ の間に，リファレンス演算器では関係 $R(x_1, \cdots, x_n, y_1, \cdots, y_m)$ が，実装演算器では関係 $I(x_1, \cdots, x_n, y_1, \cdots, y_m)$ が成り立っている時，論理式

$$\begin{aligned}&\forall x_1 \cdots \forall x_n \ \forall y_1 \cdots \forall y_m \\ &\quad (I(x_1, \cdots, x_n, y_1, \cdots, y_m) \\ &\qquad \rightarrow R(x_1, \cdots, x_n, y_1, \cdots, y_m))\end{aligned}$$

が真であれば実装は正しい[8]．

ここでたとえば，リファレンスにおいて出力 y_k（$1 \leq k \leq m$）が入力を引数とする論理関数 f_k で定義されるなら，R は”$y_1 = f_1(x_1, \cdots, x_n) \wedge \cdots \wedge y_m = f_m(x_1, \cdots, x_n)$” である．

ここまでは一般の演算器検証と同様だが，AES 演算器では，以下の工夫を併用する：

(A) SubBytes，逆元演算，TBox は，テーブル（真理値表）実装される場合がある．テーブルを検証するには，入出力等価な論理式を検証処理系で機械的に作成する．これは正極性ダビオ展開によって可能である[9]．

[8] $a \rightarrow b$ は $(\neg a) \mid b$ と等価．包含でなく等価性を判定する場合は，$I(\cdots) = R(\cdots)$ を検証条件とすればよい．

[9] FDD[3] を BDD[2] と同様に構成し，各ノードを

(B) AES field と異なる体や基底を使った実装に対しては，同型写像 δ と δ^{-1} の回路実装も一緒に用意した上で，

- 同型写像を実装につなぎ，入力と出力を AES field に読み替えて検証することと，
- $\forall X \in$ AES field; $\delta(\delta^{-1}(X)) = X$ を検証すること（演算前後でなく，AES プライマリ入出力で 1 度だけ変換をする場合）

が必要である．
(C) サイドチャネル対策としてランダムマスクを施した SubBytes や逆元演算については，任意のマスク値に対し演算結果が正しいことも合わせて検証する．たとえば，「任意のマスク値 $M\in$ AES field とデータ値 $X\in$ AES field に対し，MaskedSBox($M+X$,M)=Sbox(X)+M となること」を検証条件とする．

3.4 検証条件の PPRM 変換と真偽判定

検証条件の真偽を，PPRM 形式へ変換することにより機械的に判定する．PPRM は積項を XOR で結合したカノニカル・フォームであるが，検証条件は全称記号 $\forall$ で束縛されているので，変換により 1（恒真）となれば実装は正しく，0（恒偽）になったり積項が残ったりすれば実装に誤りがある．

PPRM への変換は，表 1 のルールを適用し，論理演算を AND, XOR のみにするとともに，AND を構文木の根側へ降ろしていくことで行える．この変形は必ず停止（収束）する．

ここで，以下の留意点がある：
(1) 検証条件の真偽を調べる他の方法として，標準積和形（SOP; Sum of Products）への変形や，BDD のような決定グラフを構成する手法も使える．ただし，XOR を多用する有限体演算の検証では，それらの形式では表現規模の爆発的増加が起き，非実用的な検証時間になりやすい．XOR ベースの論理表現形式であっても，FDD のようなグラフ処理より，単純な論理式変形の方が高速になりやすい（それが本稿で PPRM 変換を用いる理由である）．BDD や FDD を適用した場合の結果は，文献 [4] を参照．
(2) 本稿では議論しないが，表 1 のルール適用順が，真偽判定にかかる実時間に非常に強い影響を与える．

4 AES 実装の形式検証結果

通常の SBox や MixColumns のみならず，図 2 に挙げた様々な最適化演算部品について，検証実験を行ってみた．表 2 に結果を示す．

検証では，前章で述べた手法に従って PPRM 形式のリファレンスをまず構築し，それと実装の論理式を合わせて検証条件を作り，検証条件を PPRM に変形して真であることを確かめた．検証時間には，リファレンス構築の CPU 時間なども含む．

概して，MixColumns のような XOR マトリックスは非常に短時間で検証できた．総当りシミュレーション等と比べれば桁違いに高速である．一方，SBox など AND-XOR が多段になった回路では少し時間が増加し，入力空間も小さいので，総当りの方が短時間で済む場合も有り得る．しかし，合成体や基底を変えた場合にはテストケース作成が問題となり，シミュレーション環境構築の手間もかかるので，実作業としては本手法の方が効率的と思われる．

また，ランダムマスク付き SBox の検証は，本稿執筆時点の PPRM 処理系では時間を要してしまった．前章で触れたとおり，検証条件を PPRM に変形する途中で，式サイズが膨らんでしまったためである．表 1 のルール適用順につき，もう少し検討を深めたい．

最後に，表 2 は全て正しい実装の事例であるが，わざと幾つかの設計ミスを入れた実装についても検証を行ってみた．正確な分類は難しいが，検証時間は誤りのタイプに依存する．積項の変数を間違えた，余計な積項が入った，といった AND-XOR 形式の範疇からあまり逸脱しないミスであれば，検証時間は大きくは変わらなかった．いっぽう，AND, XOR 以外の論理演算を混ぜてしまうようなミスでは，PPRM への変形過程で式サイズが増加し，検証時間が延びや

AND-XOR に読み替える．有限体演算には BDD のシャノン展開は向かないので [4]，ダビオ展開を用いる．

表 1: 論理式を PPRM へ変形するために用いるルール

適用前	適用後
$\forall(var1 = exp_1 \& f(var1))$	$\forall(f(exp_1))$
$exp_1 = exp_2$	$\neg(exp_1 \oplus exp_2)$
$exp_1 \rightarrow exp_2$	$(\neg exp_1) \mid exp_2$
$exp_1 \mid exp_2$	$\neg((\neg exp_1)\ \&\ (\neg exp_2))$
$\neg exp_1$	$exp_1 \oplus 1$
$(exp_1 \oplus exp_2)\ \&\ exp_3$	$(exp_1\ \&\ exp_3) \oplus (exp2\ \&\ exp_3)$
$exp_1\ \&\ exp_1$	exp_1
$exp_1 \oplus exp_1$	0
$exp_1 \oplus 0$	exp_1
$0 \oplus 1$	1

表 2: AES 演算部品の検証結果

演算部品	入力 bit 数（実装）	出力 bit 数（実装）	実装規模（&, ⊕ 総数）	判定時間（ミリ秒）
Table SBox (Enc)	8	8	4025	40
合成体 SBox (Enc)[18]	8	8	136	166
合成体逆元 [18]	8	8	119	131
合成体 SBox (Enc+Dec)[18]	8	8	296	2,173
MixColumns	32	32	108	10
InvMixColumns	32	32	202	14
MixColuimns + InvMixColumns	32	64	202	24
合成体 InvMixColumns + InvAffine [18]	32	64	237	62
同型写像 [18]	8	8	12〜14	< 1
ランダムマスク付 SBox（DPA 対策）[22, 23]	16 (マスク 8)	8	480	802,637
ランダムマスク付 MixColumns（DPA 対策）[22, 23]	64	32	368	46

Core 2 Duo@3GHz，3G バイトメモリの PC を使用

すい傾向があった.

5 おわりに

本稿では AES 回路で用いる部品の形式検証手法について述べた．PPRM への論理式変形はシンプルな手法だが，AES 検証に向いた方法と思われる．また，有限体演算では正しいリファレンスの構成が厄介だが，ボトムアップ構築の方法についても示した．今後，検証時間の短縮や他手法とのハイブリッド化などについて検討を進めたい.

謝辞

本研究の実施にあたり活発なご議論とご示唆をいただいた，東北大学大学院情報科学研究科の本間尚文准教授と齋藤和也氏に深く感謝いたします.

参考文献

[1] National Institute of Standards and Technology (NIST): Advanced Encryption Standard (AES), *FIPS Publication 197*, http://csrc.nist.gov/encryption/aes/index.html (2001).

[2] R.E. Bryant, "Graph-based algorithms for Boolean function manipulations," IEEE Trans. on Computers, Vol. C-35, No. 8, pp. 677-691, 1986.

[3] U. Kebshull and W. Rosenstiel, "Efficient Graph-Based Computation and Manipulation of Functional Decision Diagrams," European Design Automation Conf. '93, pp. 278-282, 1993.

[4] Sumio Morioka, Yasunao Katayama and Toshiyuki Yamane, "Towards Efficient Verification of Arithmetic Algorithms over Galois Fields $GF(2^m)$," 13th Conference on Computer Aided Verification (CAV'01), LNCS Vol.2102, pp.465-477, 2001.

[5] 森岡澄夫, "ガロア体上の誤り訂正符号の設計検証法", 第 15 回回路とシステム軽井沢ワークショップ, pp.275-280, 2002.

[6] Jae-Gon Lee and Chong-Min Kyung, "PrePack: Predictive Packetizing Scheme for Reducing Channel Traffic in Transaction-Level Hardware/Software Co-Emulation," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, pp.1935-1949, 2006.

[7] D.Mukhopadhyay, G.Sengar and D.R.Chowdhury, "Hierarchical Verification of Galois Field Circuits," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol.26, No.10, pp.1893-1898, 2007.

[8] Jinpeng Lv, P.Kalla and F.Enescu, "Verification of composite Galois field multipliers over $GF((2^m)^n)$ using computer algebra techniques," IEEE International High Level Design Validation and Test Workshop (HLDVT), pp.136-143, 2011.

[9] Jinpeng Lv, P.Kalla and F.Enescu, "Efficient Gröbner basis reductions for formal verification of galois field multipliers," Design, Automation and Test in Europe Conference and Exhibition (DATE), pp.899-904, 2012.

[10] Jinpeng Lv and P.Kalla, "Formal Verification of Galois Field Multipliers Using Computer Algebra Techniques," 25th International Conference on VLSI Design (VLSID), pp.388-393, 2012.

[11] A.Slobodova, "Formal Verification of Hardware Support for Advanced Encryption Standard," Formal Methods in Computer-Aided Design (FMCAD'08), 2008.

[12] E.W.Smith and D.L.Dill, "Automatic Formal Verification of Block Cipher Implementations," Formal Methods in Computer-Aided Design (FMCAD'08), 2008.

[13] H.Post and C.Sinz, "Proving Functional Equivalence of Two AES Implementations Using Bounded Model Checking," International Conference on Software Testing Verification and Validation (ICST'09), pp.31-40, 2009.

[14] 齋藤和也, 本間尚文, 青木孝文, "ガロア体上の算術演算回路の形式的設計とその AES 暗号プロセッサへの応用," 第 29 回 暗号と情報セキュリティシンポジウム（SCIS2012）, 4C1-4, pp.1-8, Jan 2012.

[15] K.Saito, N.Homma and T.Aoki, "A Formal Approach to Designing Arithmetic Circuits over Galois Fields Using Symbolic Computer Algebra", The 17th Workshop on Synthesis And System Integration of Mixed Information Technologies (SASIMI2012), R2-2, pp.153-158, Mar 2012.

[16] N.Homma, K.Saito and T.Aoki, "A Formal Approach to Designing Cryptographic Processors Based on $GF(2^m)$ Arithmetic Circuits," IEEE Transactions on Information Forensics and Security, Vol.7, No.1, pp.3-13, 2012.

[17] C. Paar, "A New Architecture for a Parallel Finite Field multiplication with Low Complexity Based on Composite Fields," IEEE Trans. on Computers, Vol. 45, No. 7, pp. 856-861, 1996.

[18] A. Satoh, S. Morioka, K. Takano, and S. Munetoh, "A Compact Rijndael Hardware Architecture with S-Box Optimization", ASIACRYPT 2001, LNCS Vol.2248, pp.239-254, 2001.

[19] S.Morioka and A.Satoh, "A 10Gbps Full-AES Crypto Design with a Twisted-BDD S-Box Architecture," Proc. IEEE Intl. Conf. on Computer Design 2002 (ICCD2002), pp.98-103, 2002.

[20] 根角健太, 森岡恵理, 野上保之, "$F_{(2^4)^2}$ 上の逆元計算を用いた AES 用 SubBytes 変換回路の小型化," 第 29 回 暗号と情報セキュリティシンポジウム（SCIS2012）, 1C2-1, Jan 2012.

[21] A. Halbutogullar and C.K. Koc, "Mastrovito Multiplier for General Irreducible Polynomials," IEEE Trans. on Computers, Vol. 45, No.5, pp. 503-518, 2000.

[22] 森岡澄夫, 秋下徹, "合成体を用いた AES S-Box 回路に対する DPA 攻撃," コンピュータセキュリティシンポジウム 2004 (CSS2004), pp.679-684, 2004.

[23] A.Moradi and O.Mischke, "Glitch-free implementation of masking in modern FPGAs," IEEE International Symposium on Hardware-Oriented Security and Trust (HOST), pp.89-95, 2012.