SCIS 2013 The 30th Symposium on Cryptography and Information Security
Kyoto, Japan, Jan. 22 - 25, 2013
The Institute of Electronics,
Information and Communication Engineers

セキュアな鍵格納とチャレンジレスポンス認証のための統合コプロセッサアーキテクチャ

Unified Coprocessor Architecture for Secure Key Storage and Challenge-Response Authentication

清水 孝一* Koichi Shimizu　鈴木 大輔* Daisuke Suzuki　鶴丸 豊広* Toyohiro Tsurumaru　菅原 健* Takeshi Sugawara
汐崎 充† Mitsuru SHIOZAKI　藤野 毅‡ Takeshi FUJINO

あらまし グリッチ PUF およびブロック暗号の回路を用いて、セキュアな鍵格納とチャレンジ・レスポンス認証に必要な機能を効率良く統合したコプロセッサのアーキテクチャを提案する。グリッチ PUF はグリッチを発生させる目的でランダムロジックを使用していることから、提案するアーキテクチャはブロック暗号回路を中心とし、そのラウンド関数をグリッチ PUF におけるランダムロジックとして共有できるように設計する。具体例としてグリッチ PUF と AES を用いた回路構成を示し、FPGA による評価結果を示す。また、上記と同一の回路を用いた物理乱数生成器を提案し、評価結果として、乱数検定の目的で標準的に使用されている NIST SP800-22 および Diehard に合格することを示す。

キーワード Physical Unclonable Function，FPGA, Spartan-3A, Spartan-6, Implementation，Evaluation

1 はじめに

Physical Unclonable Function（PUF）[1, 2] は、ASIC や FPGA などの汎用 LSI 上で耐タンパ性を実現する方式として注目される技術である。PUF は、入力されたチャレンジに対してレスポンスを出力する関数であり、PUF が実装されている人工物の物理特性に依存してレスポンスが決定する。物理特性は製造ばらつきによって生じるため、全く同じ物理特性を持つ人工物を複製することは困難である。ノイズを含むデータから安定的な情報を抽出する Fuzzy Extractor（FE）と組み合わせることにより、PUF は複製困難なデバイス固有鍵を生成する目的で利用可能となる。LSI で PUF を実現する方式は、(1) メモリセルの特性を利用する SRAM PUF などの方式と、(2) 回路遅延のばらつきを利用する Arbiter PUF などの方式に分けられる。LSI 上で PUF を実現する方式や PUF を用いた鍵生成に関しては、これまでに多くの研究がなされている [2, 5, 6, 7, 8, 9, 10, 11, 12, 13]。

ただし、PUF と Fuzzy Extractor のみでセキュリティシステムに必要な全ての機能をカバーできるわけではない。例えば、一般的なチャレンジ・レスポンス認証によるデバイス認証では、鍵付き一方向性ハッシュ関数のような、ブロック暗号処理が必要であり、またチャレンジの生成に乱数生成が必要である。この差異を埋めるため、本論文ではグリッチ PUF をベースとして、セキュアな鍵格納とチャレンジ・レスポンス認証に必要な機能を効率的に統合したコプロセッサアーキテクチャを提案する。グリッチ PUF は回路遅延を利用した PUF であり、PUF 出力を生成するタイミングに制約が無い点と、一般的な回路設計の手法で実装可能な点が特長である。グリッチ PUF がランダムロジックを利用していることから、提案するアーキテクチャはブロック暗号回路を中心に据え、そのラウンド関数をグリッチ PUF でも利用可能な設計とする。また、グリッチ PUF と Fuzzy Extractor を組み合わせた鍵生成機能に加え、本論文ではグリッチ PUF を

* 三菱電機株式会社 情報技術総合研究所 〒 247-8501 神奈川県鎌倉市大船 5-1-1. Mitsubishi Electric Corporation, Information Technology R & D Center, 5-1-1 Ofuna Kamakura, Kanagawa, 247-8501, Japan.
† 立命館大学総合理工学研究機構 〒 525-8577 滋賀県草津市野路東 1-1-1. Research Organization of Science & Engineering, Ritsumeikan University, 1-1-1 Nojihigashi, Kusatsu, Shiga, 525-8577, Japan.
‡ 立命館大学大学院理工学研究科 〒 525-8577, 滋賀県草津市野路東 1-1-1, Graduate School of Science and Technology, Ritsumeikan University, 1-1-1 Nojihigashi, Kusatsu, Shiga, 525-8577, Japan.

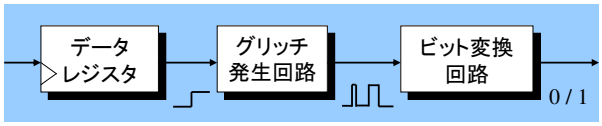

図 1: GPUF の基本構成

用いた物理乱数生成器を同一の回路を用いて実現する。これは、グリッチ PUF のエラーレートがランダムロジックの複雑さに応じて上昇する性質を利用している。

1.1 本論文の貢献

本論文は、セキュアな鍵格納とチャレンジ・レスポンス認証のためのコプロセッサを小規模回路で実現するためのアーキテクチャを提案する。基本アイデアは暗号化・鍵生成・乱数生成の回路を共有化することである。この目的に適したグリッチ PUF を用いた乱数生成器を提案する。コプロセッサの構成例として AES とグリッチ PUF を用いた回路構成を示し、製造プロセスが異なる 2 種類の FPGA に対する評価結果を報告する。乱数生成器は、標準的な乱数検定プログラムである NIST SP800-22 および Diehard の両検定に合格する。また PUF の特性として、製造プロセスが微細化すると信頼性が下がりユニーク性が上がる傾向が分かった。

2 GPUF を用いた乱数生成器

乱数生成は暗号システムを運用する上で必要な機能の 1 つであり、初期化ベクトルやセッション鍵、チャレンジ・レスポンス認証に用いる乱数列等を生成するため使用される。実装性やコストなどの観点から、アナログ回路を必要とせずデジタル回路のみで実現可能な乱数生成器がこれまでに研究されてきた [15, 16, 17]。さらに、文献 [18, 19] では、回路の共有構造により PUF と乱数生成の 2 つの機能を効率的に実装する方式が提案されている。

本論文は、PUF と乱数生成に加え、さらに暗号回路も共有構造で効率的に実現することを目的とする。まず、グリッチ PUF を利用した乱数生成器を提案する。本節ではグリッチ PUF を利用した乱数生成器の原理を説明し、次節で具体的な回路構成を与える。

2.1 GPUF の特徴

グリッチ PUF (GPUF) [12, 13] は、論理回路を構成するゲート間の遅延差によって生じるグリッチを利用する PUF である。図 1 に示す GPUF の基本構成において、データレジスタの制御によりグリッチ発生回路への入力を変化させると、その出力にグリッチが生じる。発生したグリッチはその形状に応じて、たとえば立ち上がりエッジの個数が偶数または奇数であるかによって、0 または 1 のビットに変換される。

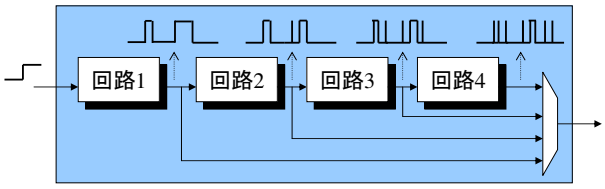

図 2: 選択可能なグリッチ発生回路

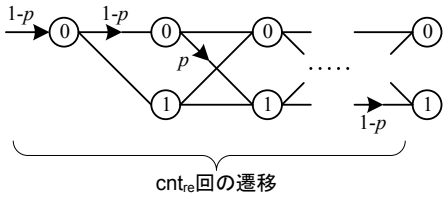

図 3: 内部状態 r_k の遷移

グリッチ発生回路が複雑になるほどグリッチがより多く発生する。その結果、グリッチ PUF のレスポンスから得られる情報量が増加するが、その反面エラーレートも向上する。従って、Fuzzy Extractor を用いた鍵生成等、PUF の通常の用途においては情報量とエラーレートのバランスが良いグリッチ発生回路を設計する必要がある。文献 [12, 13] は、情報量とエラーレートをシミュレーションで事前評価する仕組みを提案し、グリッチ発生回路として AES の SubBytes を使用したグリッチ PUF に対して事前評価結果と実機評価が一致することを示した。これにより、グリッチ発生回路の事前評価を可能とした。

2.2 乱数生成を考慮したグリッチ発生回路

一方、もしエラーレートが 50%近くになれば、GPUF のレスポンスを乱数として利用できる可能性がある。この目的に対しては、より複雑なグリッチ発生回路を使用するのが良く、通常の PUF とは異なるグリッチ発生回路が要求される。

従って、我々は PUF と乱数生成を同一の回路で実現するため、グリッチ発生回路を選択可能な構成にする。その基本原理を図 2 に示す。

図 2 では 4 つの回路を直列に繋ぎ、その中間出力を選択可能にすることで、4 種類のグリッチ発生回路を実現している。構成要素である回路 1〜4 は、互いに異なる回路であっても、全て同じであっても良いが、回路を多く通るほどグリッチ発生回路として複雑になり、グリッチがより多く発生する。

2.3 エラーレートの積算

以上の構成でエラーレートが高くなるようなグリッチ発生回路の選択が可能となるが、エラーレートが 50%近

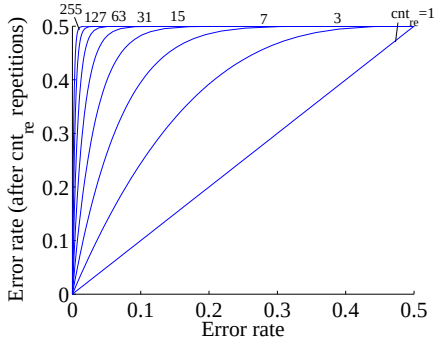

図 4: エラー積算効果

くになることをあらかじめ保証するのは困難である。そこで、同一のチャレンジに対するレスポンス生成の繰り返しによりエラーレートを積算する仕組みを設ける。次式で内部状態 r_k を定義し、$\mathsf{cnt_{re}}$ 回のレスポンス生成繰り返し後の値 $r_{\mathsf{cnt_{re}}+1}$ を最終的な出力とする。

$$r_1 := 0$$
$$r_{k+1} := r_k \oplus b_k$$

ここで $r_1 = 0$ は内部状態の初期値であり、また b_k は k 回目のレスポンス生成で得られたレスポンスビットを表す。

正しいレスポンスビットを 0 とし、エラーレートを p とすると、各 k に対し確率 $1-p$ で $b_k = 0$、確率 p で $b_k = 1$ となる。従って、上式から確率 $1-p$ で $r_{k+1} = r_k$、確率 p で $r_{k+1} = r_k \oplus 1$ となる。つまり r_k は、確率 $1-p$ で値を保ち、確率 p で値を反転させる。r_k の値の遷移を図 3 に示す。

r_k の値は合計 $\mathsf{cnt_{re}}$ 回遷移するが、その最終的な値はエラーが起こった回数が偶数であれば 0、すなわち正しいレスポンスビットの値になる。逆に奇数であれば誤った値である 1 となる。従って、$\mathsf{cnt_{re}}$ 回の繰り返しによるエラー積算後のエラーレートは、エラーが奇数回起こる確率として計算できる。

$$\sum_{k=0}^{\lfloor \mathsf{cnt_{re}}/2 \rfloor} \binom{\mathsf{cnt_{re}}}{2k+1} p^{2k+1}(1-p)^{\mathsf{cnt_{re}}-(2k+1)}.$$

正しいレスポンスビットが 1 の場合も同様である。上式によるエラー積算効果を表すグラフを図 4 に示す。

例えばエラーレートが 10%程度であれば、繰り返し回数 31 でエラーレートを 50%近くに高められることが分かる。

2.4 GPUF の繰り返し処理

エラーレートの精算に関連して、GPUF がエラーレート低減のために行う繰り返し処理について説明する。同じチャレンジに対するレスポンスビット生成を $\mathsf{cnt_{re}}$ 回繰り返し、そのうち 1 が生成される回数を n とする。もしエラーが全く起きないと仮定すると、$n = 0$ ならばレスポンスビットは 0 であり、$n = \mathsf{cnt_{re}}$ ならばレスポンスビットは 1 であると決定できる。しかし実際にはエラーが起きるため、安定性の閾値 $\mathsf{th_{err}}$ によって許容するエラーの回数を決め、$n = 0, \cdots, \mathsf{th_{err}}$ ならばレスポンスビットを 0 とし、$n = (\mathsf{cnt_{re}} - \mathsf{th_{err}}), \cdots, \mathsf{cnt_{re}}$ ならばレスポンスビットを 1 とする。例えば、繰り返し回数 $\mathsf{cnt_{re}} = 7$、閾値 $\mathsf{th_{err}} = 2$ のとき、

- $n = 0, 1, 2$: レスポンスビットは 0
- $n = 3, 4$: 値を決定できない不安定な状態
- $n = 5, 6, 7$: レスポンスビットは 1

としてレスポンスビットの値、および、安定性が決定される。

第 2.3 節で述べたエラーレート低減のための繰り返し処理と、本節で述べたエラーレート積算のための繰り返し処理は、同一の回路によって実現可能である。第 3.4 節で説明する。

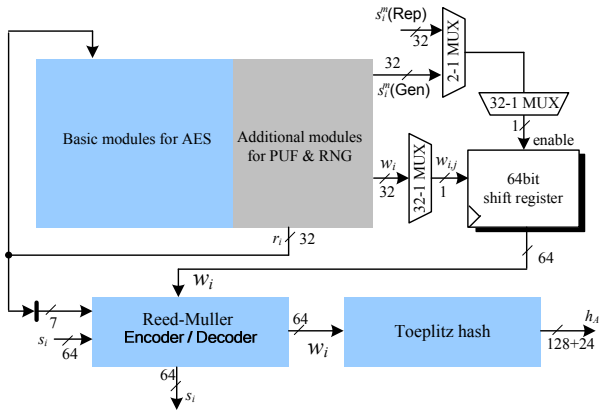

図 5: 統合コプロセッサのアーキテクチャ

3 統合コプロセッサ

我々が文献 [14] で提案した GPUF を用いた鍵生成回路に、前節で説明した乱数生成機能を付加し、さらに暗号回路を統合する。これにより、セキュアな鍵格納とチャレンジ・レスポンス認証を実現する統合コプロセッサを実現する。

3.1 アーキテクチャ

統合コプロセッサのアーキテクチャを図 5 に示す。鍵生成回路は、我々が文献 [14] で提案したものをベースとし、図 5 の下部、Reed-Muller Encoder / Decoder および Toeplitz hash については、文献 [14] と同一の構成である。

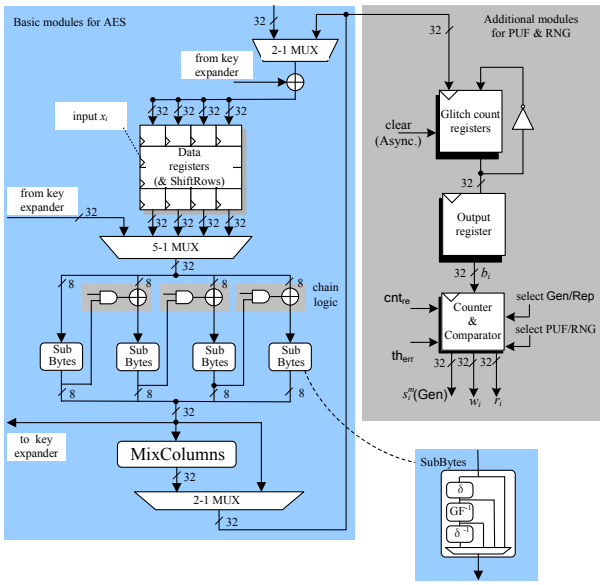

図 6: AES 回路と GPUF 回路

3.2 AES 回路と GPUF 回路の設計

図 5 の上部、AES 回路と GPUF 回路の設計について説明する。文献 [12, 13] で提案されたグリッチ PUF は、AES の SubBytes をグリッチ発生回路として用い、Spartan-3A 上での評価の結果、PUF として良好な性質を示している。そこで、AES の SubBytes をグリッチ発生回路のベースとする。一方、前節で説明した乱数生成の目的のため、さらに複雑なグリッチ発生回路も選択可能とする。逆に、エラーレートをより低く抑える場合も考慮し、SubBytes より単純なグリッチ発生回路も選択可能とする。AES の実装に関しては多数の既存研究が知られているが、本論文では文献 [20] の実装を用いる。

以上に基づき、AES の SubBytes を中心とする共有構造を持つ AES 回路と GPUF 回路を構成する。構成した回路を図 6 に示す。図の左側がベースとなる AES 回路であり、右側が GPUF 用の追加回路である。AES のデータパスにセレクタを導入し、行う処理によってデータパスを切り替えられるようにしている。また、SubBytes より単純なグリッチ発生回路も選択可能にするため、合成体実装の SubBytes における各段階からの出力を選択可能にしている。

3.3 PUF 処理

図 6 の回路の動作を説明する。PUF の処理を行う場合、AES のデータレジスタを GPUF へのチャレンジ入力として使用する。GPUF では、グリッチ発生回路への入力の変化パターンがチャレンジとなる。これに対し、AES に対する平文および鍵が、チャレンジの初期値として機能する。そしてデータレジスタ後の 5-1 セレクタを切り替えることによって、初期値からの信号遷移が発生し、その後段の回路でグリッチが発生する。

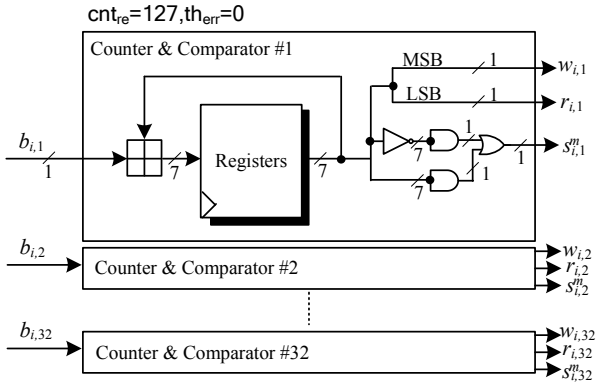

図 7: Counter & Comparator の構成例

追加回路中の Glitch count register は、発生したグリッチ信号をクロック入力とするトグルフリップフロップ（TFF）である。TFF はクロックに立ち上がりエッジが入力されるたびに値が反転する FF なので、グリッチ信号をそのクロックに入力することにより、グリッチ信号に含まれる立ち上がりエッジの個数が偶数であるか奇数であるかに従って、0 または 1 のビットを生成できる [13]。Glitch count register はチャレンジ入力前にリセットされ、チャレンジ入力によって発生したグリッチを取り込んでビットを生成する。

次に、Counter & Comparator によって、エラーレート低減のための繰り返し処理が行われる。例として、$\mathsf{cnt_{re}} = 127$ および $\mathsf{th_{err}} = 0$ の場合の回路構成を図 7 に示す。必要なレジスタのサイズは、繰り返し回数 $\mathsf{cnt_{re}}$ の値によって決まる。

3.4 乱数生成処理

乱数生成を行う場合、GPUF へのチャレンジは固定する。本論文では固定チャレンジとして、オール 0 からオール 1 への変化を使用する。グリッチの発生パターンはチャレンジに依存するが、オール 0 からオール 1 への変化のように、十分大きな変化であればチャレンジは任意である。

第 2.2 節で述べた通り、乱数生成を行うためには GPUF のエラーレートを高める必要があるので、より複雑なグリッチ発生回路を選択する。図 6 の回路において、隣り合う SubBytes 間のチェーンロジックを有効化しフィードバックパスを作ることで、論理段数を高めた複雑なグリッチ発生回路を実現できる。また、SubBytes の後段に位置する MixColumns のデータパスを選択することでもグリッチ発生回路の複雑化が可能である。

ただし、第 2.3 節で述べた通り、グリッチ発生回路の複雑化のみでエラーレートを 50%近くに高められるかは保証できないため、繰り返し処理によるエラーの積算を行う。通常の PUF 処理でエラーレート低減の目的で使用する Counter & Comparator を、エラー積算の目

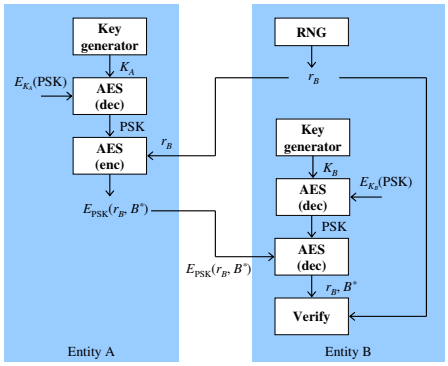

図 8: チャレンジ・レスポンス認証のフロー

的でも用いることができる。図 7 の回路構成において、Registers の最下位ビットに注目すると、$b_{i,j} = 1$ が入力されると値が反転し、$b_{i,j} = 0$ が入力される値を保持することが分かる。これは、図 3 で説明したエラーの積算に対応する挙動であるので、Registers の最下位ビットを乱数ビットとして出力することができる。

4 セキュアな鍵格納とチャレンジ・レスポンス認証

提案した統合コプロセッサを用いたセキュアな鍵格納とチャレンジ・レスポンス認証のシステム構成を述べる。PSK は認証を行うエンティティ間で事前共有される鍵であり、次のようにしてセキュアに格納される。まず、統合コプロセッサの鍵生成機能で生成した鍵 K を用いて暗号化し、暗号化結果の $E_K(\mathrm{PSK})$ をフラッシュメモリに格納する。鍵生成機能は PUF ベースであるため、$E_K(\mathrm{PSK})$ を復号できるのは、K を生成したのと同一の PUF 回路を持つエンティティのみである。さらに、K と PSK は使用時にのみ存在するので、攻撃者がこれらにアクセスできる機会は非常に限られている。これによって、セキュアな鍵格納が実現されている。

次に、チャレンジ・レスポンス認証のフローを図 8 に示す。図 8 は ISO/IEC 9798-2 をベースとした単純な認証プロトコルであり、エンティティ B がエンティティ A を認証するフローを表している。まず B は、統合コプロセッサの乱数生成機能を利用して乱数 r_B を生成し A に送信する。次に、A は (r_B, B^*) を事前共有鍵 PSK で暗号化し B に返信する。ここで B^* は、反射攻撃を防止するためのオプションデータである。A が PSK を使用するためには、まず $E_{K_A}(\mathrm{PSK})$ をフラッシュメモリから読み出し K_A で復号する。K_A は統合コプロセッサの鍵生成機能によって生成される鍵である。PSK の使用後、PSK および K_A はレジスタから消去される。最後に、B は受け取ったデータを PSK で復号し、復号結果が初めの r_B と一致することを検証する。B が PSK を使用する手順は A と同様である。

表 1: パラメータ

パラメータ	概要
$\mathsf{cnt_{re}} = 1, 3, \cdots, 255$	繰り返し処理の回数
$\mathsf{th_{err}} = 0, 1, \cdots, \lfloor \mathsf{cnt_{re}}/2 \rfloor$	安定性の閾値
$\mathsf{sel} = 0, 1, \cdots, 63$	グリッチ発生回路の選択
生成ビット数 = 16384	-

表 2: 評価対象回路（抜粋）

回路番号（sel）	概要
0	MixColumns
4	s1 + MixColumns
8	s1 + s2 + MixColumns
12	s1 + s2 + s3 + MixColumns
16	SubBytes

5 性能評価

本論文で構成した GPUF 回路の性能を評価し、最適なパラメータを検討する。本論文の GPUF におけるパラメータを表 1 にまとめる。

ターゲットデバイスとして Xilinx 社の FPGA である Spartan-3A および Spartan-6 を使用し、それぞれ 6 石を用いて評価を行った。動作環境の変化も含めた性能を測定するため、恒温槽により温度を、FPGA ボード上のポテンショメータにより電圧を制御可能とした。

5.1 PUF の性能評価

合計で 17 種類のグリッチ発生回路に対して GPUF の評価を行う。一部を表 2 に示す。

s1, s2, s3 は合成体実装の SubBytes を構成する δ, GF^{-1}, δ^{-1} の各ブロックを順番に表し、s1 + s2 + s3 とは SubBytes そのもののことである。

評価指標として、PUF の性能評価で一般的に用いられる信頼性およびユニーク性を用いる。加えて、GPUF ではマスク処理によって不安定ビットが除去されるため、実際に使用可能なビットの割合をビットの生成効率として評価する以下では、ビット列 A, B に対し、$\mathsf{HD}(A, B)$ は A と B のハミング距離を、$A \cap B$ はビット毎の AND を表す。また $\mathsf{HW}(A)$ は A のハミング重みを意味する。

信頼性　信頼性は、同一のデバイス上で元に近いレスポンスを再現できることを保証する指標である。信頼性の尺度として、基準となるレスポンスと再生成したレスポンスとのハミング距離が用いられる。本論文では、ハミング距離を生成ビット数で割った値であるビットエラーレート（BER）を用いる。さらに、マスク処理によるビットの除去を含めると BER は次式で定義される。

$$\mathrm{BER} := \mathsf{HD}(W^{\mathrm{n}} \cap S^{\mathrm{n}}_{\mathrm{mask}}, W^{\mathrm{c}} \cap S^{\mathrm{n}}_{\mathrm{mask}}) / \mathsf{HW}(S^{\mathrm{n}}_{\mathrm{mask}})$$

X^{n} は常温・常電圧下 (27°C, 1.20V) におけるデータを、X^{c} はコーナーケースにおけるデータを表す。

ユニーク性　ユニーク性は、異なるデバイス上で異なるレスポンスが生成されることを保証する指標である。ユ

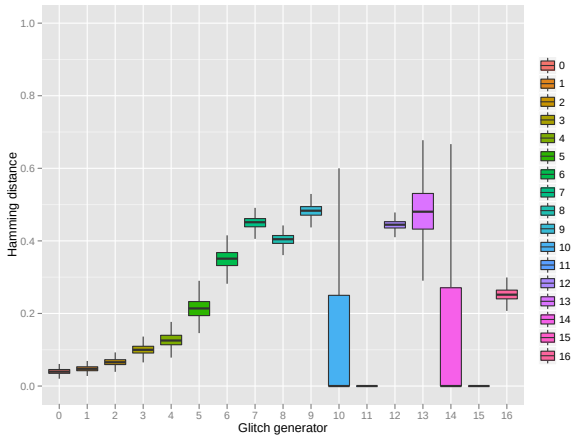

図 11: Spartan-3A におけるユニーク性

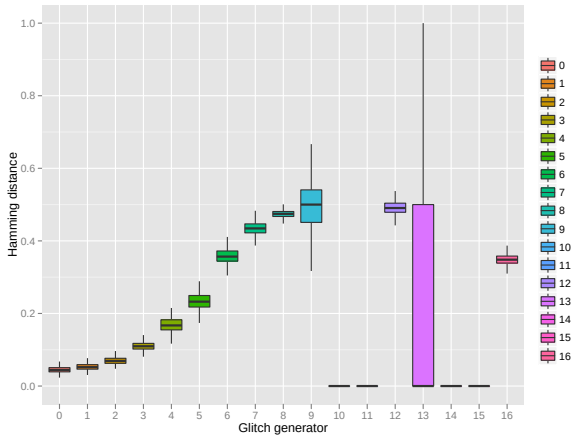

図 12: Spartan-6 におけるユニーク性

ニーク性の尺度として、異なるデバイスで生成されたレスポンス間のハミング距離が用いられ、ハミング距離が生成ビット数の 50%に近づくほどユニーク性が高い。本論文では、ハミング距離を生成ビット数に対する割合で表した FHD を用いる。GPUF のマスク処理を含め、FHD は次式で定義される。

$$\mathrm{FHD} := \mathsf{HD}(W^i \cap S^i_{\mathrm{mask}}, W^j \cap S^i_{\mathrm{mask}})/\mathsf{HW}(S^i_{\mathrm{mask}})$$

X^i は FPGA 番号 i に対するデータを表す。

生成効率　ビットの生成効率として、マスクビットが有効になる確率を評価する。

$$P_r[s=1] := \mathsf{HW}(S^{\mathrm{n}}_{\mathrm{mask}})/16384$$

5.2　評価結果

信頼性　信頼性の評価結果を図 9 および 10 に示す。FE を用いた鍵生成を現実的な回路規模で実現するため、エラーレートは高々20%程度に抑える必要がある。従って、Spartan-3A で使用可能な回路は 1〜6 および 16 に絞られ、Spartan-6 では回路 1〜5 に絞られる。なお、回路 10 などエラーレートが見かけ上 0%近くのものがあるが、これらはマスクの有効ビットがほぼ 0 のものである。このことは、図 13 によって示される。

ユニーク性　ユニーク性の評価結果を図 11 および 12 に示す。 信頼性で絞った回路から最もユニーク性の高いものを選択すると、Spartan-3A では回路 6 が、Spartan-6 では回路 5 が候補となり、それぞれ 37%および 23%程度のハミング距離となる。後者は PUF として実用することが困難な値である。

生成効率　生成効率の評価結果を図 13 に示す。ビットの生成効率が高いほど、FE を用いた鍵生成で必要となるビット数を高速に生成することができる。Spartan-3A で候補とした回路 6 では 66%程度の生成効率を確保しており、十分といえる。

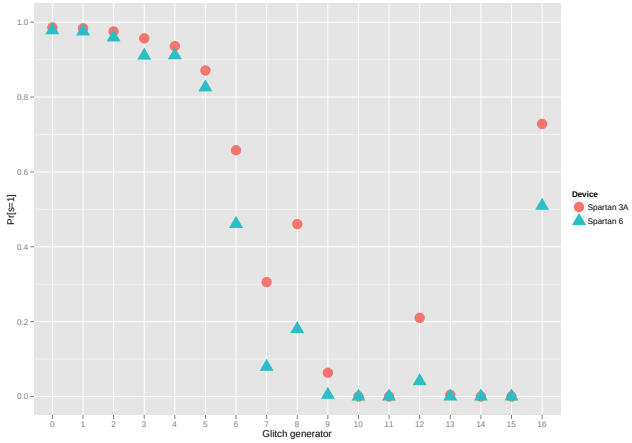

図 13: GPUF のビット生成効率

評価のまとめ　各指標による評価結果から、Spartan-3A ではグリッチ発生回路 6 を用いた GPUF が、FE を用いた鍵生成等で必要な性能をみたせることが分かった。一方 Spartan-6 に関しては、本論文で評価したグリッチ発生回路では PUF として実用可能な性能がみたせない可能性が高く、グリッチ発生回路に関してはより詳細な検討が必要であることが分かった。

Spartan-6 上の GPUF に関しては既存研究 [21] があり、Spartan-3A 上の結果 [12, 13] と比較して電圧特性が悪化することが示されている。本論文の評価結果から、Spartan-6 上では信頼性が下がり、ユニーク性が上がる特性が分かった。原因として、90nm から 45nm への製造プロセスの微細化によりグリッチが発生しやすくなることが考えられる。

また文献 [21] では、SubBytes をグリッチ発生回路とする GPUF は FPGA 上で実現困難とされている。しかし、図 9 および図 11 で示した回路 16 の結果は、Spartan-3A 上で SubBytes を用いた GPUF が実現可能であることを示しており、ターゲットデバイスの違いを含めた議論が必要であることが分かった。

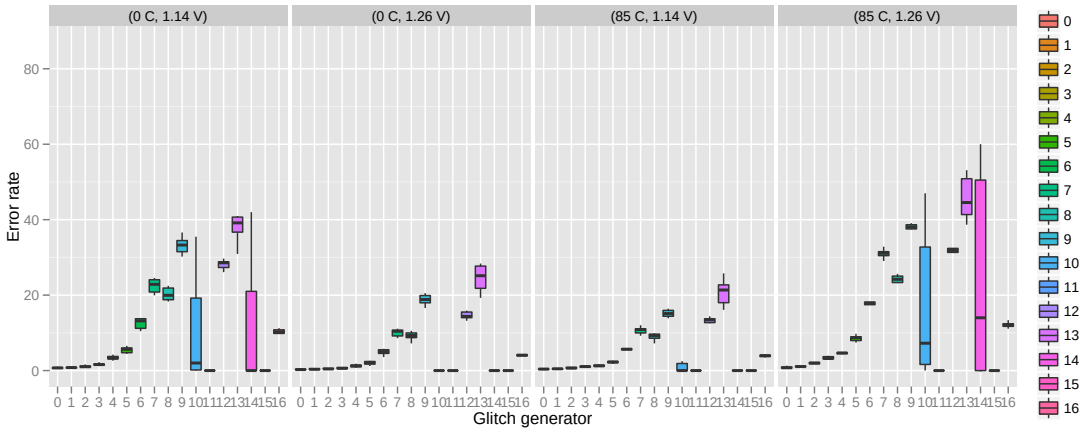

図 9: Spartan-3A における信頼性

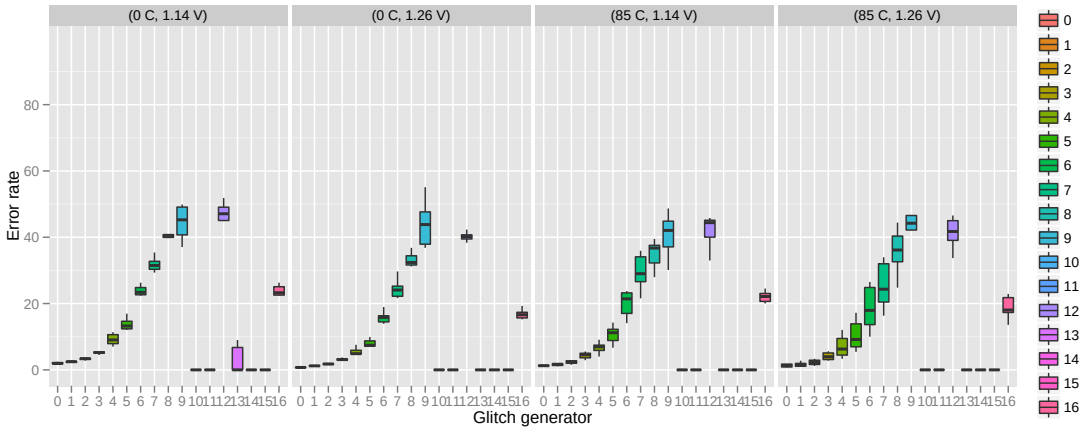

図 10: Spartan-6 における信頼性

5.3 乱数生成器の性能

ビット列の乱数性を評価するため標準的に使用される検定プログラムとして NIST SP800-22 [22] および Diehard [23] がある。本論文では、各動作条件において 1G ビットのビット列を生成し両検定で評価を行った。表 1 のパラメータは $\mathsf{cnt}_{\mathsf{re}} = 255$ および $\mathsf{th}_{\mathsf{err}} = 0$ とした。

NIST 検定では、まず 1G ビット列を 1000 個の 1M ビット列に分割し、その各 1M ビット列に対して検定を行う。各検定項目に対し、1000 回の検定が行われ、その結果 1000 個の p 値が得られる。p 値とは、検定対象のビット列が、検定項目が想定する統計的性質に合致する確率である。ただし、高い p 値が得られれば良いわけではなく、1000 個の p 値が一様に分布していることが、乱数に求められる性質である。

一方、Diehard 検定では、1 回の検定に約 80M ビットのビット列が必要となる。そこで、本論文では 1G ビットを分割して 12 個の 80M ビット列を生成し、そのそれぞれに対して検定を行った。Diehard では 1 回の検定で 220 個の p 値が出力されるので、12 回の検定により合計 2640 個の p 値が得られる。

評価の結果、GPUF を用いた乱数生成器は NIST と Diehard の両検定に合格することが分かった。検定結果の例として、常温・常電圧下における Spartan-6 上での結果を表 3 および表 4 に示す。紙面の都合上、表 3 は 3 石のみの結果を記している。また、動作環境のコーナーケースである (0°C, 1.14V)、(0°C, 1.26V)、(85°C, 1.14V)、(85°C, 1.26V) においても合格することが分かった。さらに、現時点でサンプル数は少ないものの、-40°C および 125°C の温度下においても合格することを確認している。

6 まとめ

本論文では、セキュアな鍵格納とチャレンジ・レスポンス認証のための統合コプロセッサアーキテクチャを提案した。このアーキテクチャは AES 回路を中心とした共有構造により、暗号化、PUF、乱数生成の機能を効率的に実現している。AES による暗号化、GPUF を用いた鍵生成に基づくセキュアな鍵格納、および、GPUF を用いた乱数生成によって、チャレンジ・レスポンス認証に必要な機能を提供する。性能評価の結果、ターゲットデバイスが Spartan-3A の場合は PUF として必要な性能をみたせることが分かった。しかし、Spartan-6 の場合

表 3: NIST 検定の合格率 (%)

検定項目	パラメータ	FPGA 1	FPGA 2	FPGA 3	Result
Frequency	-	98.1	99.2	98.9	Pass
Block Frequency	$M = 128$	98.5	98.8	99.3	Pass
Runs	-	99.3	99.3	98.8	Pass
Longest run	$M = 10000$	98.6	98.6	99.3	Pass
Rank	-	99.3	99.7	99.3	Pass
Non-overlapping templates	$m = 9$	97.8 - 99.9	97.9 - 99.6	98.1 - 99.7	Pass
Overlapping templates	$m = 9$	99.3	98.6	99.2	Pass
Universal	$L = 7$, $Q = 1280$	98.7	99.1	98.4	Pass
Linear complexity	$M = 500$	98.9	99.2	99.2	Pass
Serial	$m = 16$	98.9 - 99.2	99.0 - 99.2	99.2 - 99.3	Pass
Approximate entropy	$m = 10$	99.1	98.9	99.2	Pass
Cumulative sums	-	98.4 - 98.5	98.8 - 99.1	98.7 - 98.9	Pass
Random excursions	-	98.7 - 99.7	98.3 - 99.4	98.3 - 99.5	Pass
Random excursions variant	-	98.2 - 99.5	98.2 - 99.2	97.8 - 99.2	Pass

表 4: Diehard 検定の合格率 (%)

検定項目	FPGA 1	FPGA 2	FPGA 3	FPGA 4	FPGA 5	FPGA 6	Result
Diehard	98.9	99.1	98.8	99.0	98.8	99.2	Pass

は、本論文で構成した GPUF では信頼性とユニーク性のバランスを保てないことが分かり、グリッチ発生回路に関する詳細な検討が必要であることが明らかになった。また、統合コプロセッサに必要な機能として、GPUF を用いる乱数生成器を提案した。標準的な乱数検定プログラムである NIST SP800-22 および Diehard を用いて評価を行い、動作環境のコーナーケースにおいても両方に合格することを示した。

謝辞

本研究の一部は、JST-CREST「ディペンダブル VLSI システムの基盤技術」の研究の一環として実施しました。関係各位に感謝いたします。

参考文献

[1] R. S. Pappu: Physical One-way Functions. Ph.D. Thesis, M.I.T., http://pubs.media.mit.edu/pubs/papers/01.03.pappuphd.powf.pdf, 2001.

[2] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas: Silicon Physical Random Functions. Proc. of the 9th ACM Conference on Computer and Communications Security (CCS 2002), pp. 148-160, 2002

[3] Y. Dodis, M. Reyzin and A. Smith: Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data. Eurocrypt 2004, LNCS 3027, pp. 523-540, Springer-Verlag, 2004.

[4] P. Tuyls and L. Batina: RFID-Tags for Anti-Counterfeiting. CT-RSA 2006, LNCS 3860, pp. 115-131, Springer-Verlag, 2006.

[5] J. Guajardo, S. S. Kumar, G. J. S̃chrijen and P. Tuyls: FPGA Intrinsic PUFs and Their Use for IP Protection. CHES 2007, LNCS 4727, pp. 63-80, Springer-Verlag, 2007.

[6] J. W. Lee, D. Lim, B. Gassend, G. E. Suh, M. van Dijk, and S. Devadas: A Technique to Build a Secret Key in Integrated Circuits for Identification and Authentication Applications. Proc. of the IEEE VLSI Circuits Symposium, pp 176-179, 2004.

[7] G. E. Suh and S. Devadas: Physical Unclonable Functions for Device Authentication and Secret Key Generation. Proc. of the 44th annual Design Automation Conference (DAC 2007), pp. 9-14, 2007.

[8] S. S. Kumar, J. Guajardo, R. Maes, G. J. S̃chrijen and P. Tuyls: Extended Abstract: The Butterfly PUF: Protecting IP on every FPGA. Proc. of the IEEE International Workshop on Hardware-Oriented Security and Trust 2008 (HOST 2008), pp. 67-70, 2008.

[9] M. Majzoobi, F. Koushanfar and M. Potkonjak: Lightweight secure PUFs. Proc. of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD 2008), pp. 670-673, 2008.

[10] C. B̈osch, J. Guajardo, A.-R. Sadeghi, J. Shokrollahi and P. Tuyls : Efficient Helper Data Key Extractor on FPGAs. CHES 2008, LNCS 5154, pp. 181-197, Springer-Verlag, 2007.

[11] R. Maes, P. Tuyls, and I. Verbauwhede, "Low-Overhead Implementation of a Soft Decision Helper Data Algorithm for SRAM PUFs," Proc. of the 2009 IEEE International Symposium on Information Theory (ISIT 2009), pp. 2101-2105, 2009.

[12] D. Suzuki and K. Shimizu: The Glitch PUF: A New Delay-PUF Architecture Exploiting Glitch Shapes. CHES 2010, LNCS 6225, pp. 366-384, 2010.

[13] K. Shimizu and D. Suzuki: Glitch PUF: Extracting Information from Usually Unwanted Glitches. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, Vol. E95-A, No. 1, 2012.

[14] 鈴木大輔, 清水孝一, 鶴丸豊広, 菅原健, 汐崎充, 藤野毅: グリッチ PUF を用いた鍵生成, SCIS 2012, 2012.

[15] B. Sunar, W. Martin, and D. Stinson: A Provabley Secure True Random Number Generator with Built-In Tolerance to Active Attacks. IEEE Transactions on Computers, Vol. 56, No. 1, pp. 109-119, 2007.

[16] M. Dichtl and J. Dj. Golic: High-Speed True Random Number Generation with Logic Gates Only. CHES 2007, LNCS 4727, pp. 45-62, 2007.

[17] K. Wold and C. H. Tan: Analysis and Enhancement of Random Number Generator in FPGA Based on Oscillator Rings. Proc. of the International Conference on Reconfigurable Computing and FPGAs, pp. 385-390, 2008.

[18] C. W. O'Donnell, G. E. Suh, and S. Devadas: PUF-Based Random Number Generation. Technical Report 481, MIT CSAIL, 2004. Available at http://csg.csail.mit.edu/pubs/memos/Memo-481/Memo-481.pdf

[19] A. Maiti, R. Nagesh, A. Reddy, and P. Schaumont: Physical Unclonable Function and True Random Number Generator: a Compact and Scalable Implementation. GLSVLSI 2009, Proc. of the 19th ACM Great Lakes symposium on VLSI, pp. 425-428, ACM, 2009.

[20] A. Satoh, S. Morioka, K. Takano, and S. Munetoh: A Compact Rijndael Hardware Architecture with S-Box Optimization. ASIACRYPT 2001, LNCS 2248, pp. 239-254, Springer-Verlag, 2001.

[21] D. Yamamoto, G. Hospodar, R. Maes and I. Verbauwhede: Performance and Security Evaluation of AES S-Box-based Glitch PUFs on FPGAs. SPACE 2012, LNCS 7644, pp. 45-62, Springer-Verlag, 2012.

[22] NIST Special Publication 800-22: A Statistical Test Suite for Random and Pseudorandom Numbers. 2000.

[23] G. Marsaglia: Diehard Battery of Tests of Randomness. http://stat.fsu.edu/pub/diehard/