

SAKURA-G のクロックグリッチによるフォールト攻撃実験

松林雅人¹ 石井潤¹ 安田正弘² 佐藤証¹

概要: SAKURA-G は暗号用と制御用に二つ FPGA Spartan-6 を有した, 暗号回路の物理攻撃評価用のプラットフォームである. FPGA が備えている DCM と PLL を用いたクロックグリッチ生成回路を実装し, クロックグリッチの挿入位置を細かく変化させながら AES 暗号回路の誤動作のパターンを詳細に解析した. その結果, 回路のクリティカルパスとグリッチの挿入位置との関係から, DFA に有用な少ないバイト数の誤りを高い再現性で発生させることが可能であることが示された. DFA では誤りのバイト数が増えるに従って計算すべき組み合わせの数が増大し, それに従って秘密鍵の導出に必要な計算量が急速に増大する. しかし攻撃ではなく, 秘密鍵を知っている状態での脆弱性評価であれば, そのような組み合わせを考慮する必要はなく, どのタイミングでクロックグリッチを入れると DFA が可能な誤りを生成することができるかといった解析が可能である. そのようないわゆるホワイトボックス評価についても検討を行った.

Fault Attack Experiment using Clock Glitch on SAKURA-G

MASATO MATSUBAYASHI¹ JUN ISHII¹ MASAHIRO YASUDA²
AKASHI SATOH¹

1. はじめに

近年, 暗号をハードウェアやソフトウェアで実装した“暗号モジュール”の実装上の物理的な脆弱性を突く攻撃が脅威となっている. 特に暗号モジュールに外部からノイズ等を混入して誤動作を誘発し, 秘密情報の漏洩を引き起こすフォールト攻撃[1]が注目されている. フォールト攻撃は, 数回の暗号化処理に対する誤動作で秘密情報を取得可能な場合もある非常に強力な攻撃法である. 暗号処理中の電力や電磁波を数千~数十万取得して統計解析を行うサイドチャネル攻撃[2]において, 安全性評価環境として広く普及した SASEBO-GII[3]ボードは, フォールト攻撃対策評価を前提としていない. 制御用と暗号用の二つの FPGA を実装しているが, 制御用は暗号用に比べ処理の遅い FPGA を用いているため, 正確なタイミングでクロックにノイズを混入する等の誤動作誘発は極めて困難であった.

そこで筆者らが開発した SAKURA-G[4]ボードは, フォールト攻撃への利用も考慮して二つの FPGA を同性能の Spartan-6 とした. 本論文では, クロックにノイズを混入するフォールト回路を SAKURA-G と SASEBO-GII の制御用 FPGA 上に実装し, 両者のフォールト攻撃実験を比較することで SAKURA-G の有用性を示す. また, 入力データと秘密鍵の求まりやすさの関連性について考察する.

2. フォールト攻撃

フォールト攻撃は, 暗号処理中のモジュールに誤動作を誘発させ, 内部の秘密情報を漏えいさせる攻撃手法である. 誤動作の誘発には, レーザーや電磁波を外部から照射した

り, 電源やクロックにノイズを混入する手法などがある. 本論文のフォールト攻撃は, クロックにノイズを混入することでエラーを含む暗号文を出力させ, 差分故障解析 (DFA: Differential Fault Analysis) [1]によって暗号の秘密鍵を導出するものである. そこでは, いかに正確なタイミングでノイズを混入し, 解析可能なエラーを含む暗号文を出力させることができるかが重要となる.

3. クロックグリッチ

エラーを誘発させるため, クロックに混入したパルス状のノイズをクロックグリッチと呼び, 以下ではグリッチと呼ぶ. 一般的な同期式回路では, クロックの立ち上がりまたは立ち下がりエッジで組み合わせ回路による演算結果がレジスタにラッチされる. 立ち上がりに同期して動作する場合, 演算が終わる前にグリッチを混入すると, それによる不正な立ち上がりで演算途中の結果がレジスタに取り込まれてしまいエラーが発生する. また, どのタイミングでグリッチを混入するかによって, エラーパターンは異なることになる.

グリッチは外部から混入させる方法と, オンボードで発生させる方法とがある. 後者は信号発生器が不要でクロックとノイズの同期がとりやすいという利点がある. そこで本論文では, 制御用 FPGA でクロックにノイズを混入するオンボードグリッチ回路を実装する. システムクロックから異なる位相でシフトさせた2つのクロックを生成し, 一方の位相シフトクロックを反転して論理積を取ることでグリッチを得る. そしてグリッチとシステムクロックの排他的論理和を取り生成したクロックを暗号用 FPGA に供給し

¹ 電気通信大学大学院情報理工学研究科

² 株式会社キャンパスクリエイト

たグリッチ混入のタイミングチャートを図1に示す。

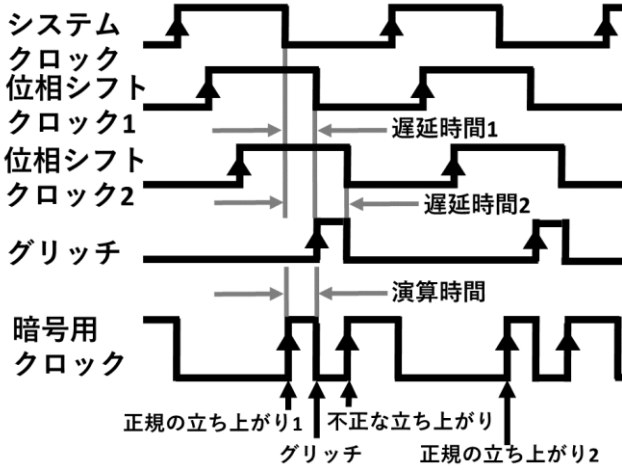


図1 グリッチ生成タイミングチャート

SAKURA-G では、48MHz のオシレータ出力から、1.5MHz ～24MHz のシステムクロックと暗号用 FPGA クロック、24MHz の USB クロックが生成されている。このクロック生成回路にノイズを混入する回路を組み込む。2 つの位相シフトクロックの生成には Xilinx の IP コアである DCM (Digital clock Manager) を 2 つ利用する。位相シフトクロックが揺らいで、ノイズ混入のタイミングの精度が下がるのを防ぐため、DCM を駆動する前にジッタをクリーンアップする PLL (Phase Locked Loop) [5] を使用する。PLL は DCM の信号の安定を待ってから出力を行うため、ジッタによる遅延が生じない。

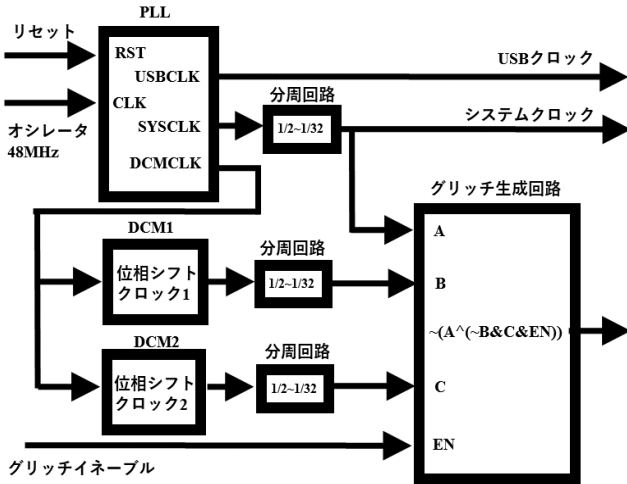


図2 クロック生成回路の概略ブロック

PLL から出力されたジッタをクリーンアップした 48MHz のシステムクロックは、2 つの DCM によって位相がシフトされる。DCM1 がグリッチの開始位置、DCM2 が終了位置の位相シフトとなる。その後、分周回路で動作周波数を 1.5MHz に落とし、最後に、システムクロックと 2 つの位相シフトクロックを論理演算してグリッチを発生させる。そして、攻撃対象とする共通鍵暗号の国際標準である AES (Advanced Encryption Standard) [6] の 10 ラウンドの処理において、特定のラウンドにのみグリッチを挿入するため

イネーブル信号を作りグリッチとの論理積を取る。

位相シフトの遅延時間は、DCM の仕様で、0～255 の値で設定し、設定幅はクロック幅の 255 分の 1 となる。DCM に入力されるクロックは SAKURA-G のオシレータが生成するクロックの周波数 48MHz のため、遅延の精度は

$$1 \div 48\text{MHz} \div 255 = 0.081699\text{nsec} \quad (1)$$

と計算できる。

また、グリッチの可変範囲は同様に DCM に入力されるクロック周波数である 48MHz を用いて

$$1 \div 48\text{MHz} = 20.833\text{nsec} \quad (2)$$

と計算できる。

SASEBO-GII でのグリッチ混入の原理は SAKURA-G と同一だが、SASEBO-GII で利用される Spartan-3 には PLL 機能が存在しないため、3 つ目の DCM を代用して 2 つの位相シフトクロックを生成している。そのため、DCM 駆動前にジッタをクリーンアップすることができず、グリッチ混入のタイミングの精度が下がると予想される。

4. DFA

本論文で用いた 128 ビットの秘密鍵の AES 暗号に対するフォールト攻撃である Moradil らによる DFA [7] を以下に示す。9 ラウンド目の *MixColumns* 処理の入力 4 バイト×4 バイト行列の 1 列目に、1～4 バイトの誤りが含まれていると仮定する。このとき正しい 4 バイト入力を A 、誤りを $\varepsilon = [e_1 \ e_2 \ e_3 \ e_4]^T$ とすると、*MixColumns* の出力は次式となり 4 バイトの誤り $\varepsilon' = [e'_1 \ e'_2 \ e'_3 \ e'_4]^T$ を含む。ここで *MixColumns*₄ の添え字 4 は 4 バイト分の処理を表している。

$$\begin{aligned} & \text{MixColumns}_4 \left(A \oplus \begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \end{bmatrix} \right) \\ &= \text{MixColumns}_4(A) \oplus \begin{bmatrix} 2 * e_1 \oplus 3 * e_2 \oplus e_3 \oplus e_4 \\ e_1 \oplus 2 * e_2 \oplus 3 * e_3 \oplus e'_2 \\ e_1 \oplus e_2 \oplus 2 * e_3 \oplus 3 * e_4 \\ 3 * e_1 \oplus e_2 \oplus e_3 \oplus 2 * e_4 \end{bmatrix} \\ &= \text{MixColumns}_4(A) \oplus \begin{bmatrix} e'_1 \\ e'_2 \\ e'_3 \\ e'_4 \end{bmatrix} \quad (3) \end{aligned}$$

10 ラウンドの正しい 4 バイト入力を $I = [I_1 \ I_2 \ I_3 \ I_4]^T$ とすると、誤りを含む入力は $[I_1 \ I_2 \ I_3 \ I_4]^T \oplus [e'_1 \ e'_2 \ e'_3 \ e'_4]^T$ と表せる。今、正しい 16 バイトの暗号文のうち解析の対象としている 4 バイトを $C = [C_1 \ C_{14} \ C_{11} \ C_8]^T$ とする。添え字が順序良く並んでいないのは、行方向への巡回シフト関数 *ShiftRows* によるものである。対応する最終 10 ラウンド目の 4 バイト分の部分鍵を $K^{10} = [K_1^{10} \ K_{14}^{10} \ K_{11}^{10} \ K_8^{10}]^T$ 、16 バイトの非線形変換 *SubBytes* の 4 バイト分を *SubBytes*₄ とすると、4 バイトの正しい暗号文出力 C は次式で表される。

$$C = \begin{bmatrix} C_1 \\ C_{14} \\ C_{11} \\ C_8 \end{bmatrix} = \text{SubBytes}_4 \left(\begin{bmatrix} I_1 \\ I_2 \\ I_3 \\ I_4 \end{bmatrix} \right) \oplus \begin{bmatrix} K_1^{10} \\ K_{14}^{10} \\ K_{11}^{10} \\ K_8^{10} \end{bmatrix} \quad (4)$$

また、誤りを含む暗号文（以降、フォールト暗号文とよぶ） C^* は次式となる。

$$C^* = \begin{bmatrix} C_1^* \\ C_{14}^* \\ C_{11}^* \\ C_8^* \end{bmatrix} = \text{SubBytes}_4 \left(\begin{bmatrix} I_1 \\ I_2 \\ I_3 \\ I_4 \end{bmatrix} \oplus \begin{bmatrix} e_1' \\ e_2' \\ e_3' \\ e_4' \end{bmatrix} \right) \oplus \begin{bmatrix} K_1^{10} \\ K_{14}^{10} \\ K_{11}^{10} \\ K_8^{10} \end{bmatrix} \quad (5)$$

となる。フォールト暗号文 C^* に含まれる誤り ε'' は式(4)と(6)の排他的論理和をとることによって次のように求まる。

$$\begin{aligned} \varepsilon'' = \begin{bmatrix} e_1'' \\ e_2'' \\ e_3'' \\ e_4'' \end{bmatrix} &= \begin{bmatrix} C_1 \\ C_{14} \\ C_{11} \\ C_8 \end{bmatrix} \oplus \begin{bmatrix} C_1^* \\ C_{14}^* \\ C_{11}^* \\ C_8^* \end{bmatrix} \\ &= \text{Subbytes} \left(\begin{bmatrix} I_1 \\ I_2 \\ I_3 \\ I_4 \end{bmatrix} \right) \oplus \text{Subbytes} \left(\begin{bmatrix} I_1 \\ I_2 \\ I_3 \\ I_4 \end{bmatrix} \oplus \begin{bmatrix} e_1' \\ e_2' \\ e_3' \\ e_4' \end{bmatrix} \right) \quad (6) \end{aligned}$$

以上は、暗号文の1, 14, 11, 8バイト目に誤りが含まれている場合であるが、5, 2, 15, 12バイト目、9, 6, 3, 16バイト目、13, 10, 7, 4バイト目それぞれのグループでも同様に計算でき、これらをそれぞれグループA、グループB2、グループC、グループDと呼ぶことにする。

本研究では、秘密鍵の導出には、(5)式、(6)式をそれぞれ *AddRoundKey*（鍵とのXOR）、*ShiftRows*、*SubBytes*の4バイト分の逆関数を施して排他的論理和をとって得られる ε' の(7)式を用いる。

$$\begin{aligned} \varepsilon' = \begin{bmatrix} e_1' \\ e_2' \\ e_3' \\ e_4' \end{bmatrix} &= \text{SubBytes}^{-1} \left(\begin{bmatrix} C_1 \\ C_{14} \\ C_{11} \\ C_8 \end{bmatrix} \oplus \begin{bmatrix} K_1^{10} \\ K_{14}^{10} \\ K_{11}^{10} \\ K_8^{10} \end{bmatrix} \right) \\ &\quad \oplus \text{SubBytes}^{-1} \left(\begin{bmatrix} C_1^* \\ C_{14}^* \\ C_{11}^* \\ C_8^* \end{bmatrix} \oplus \begin{bmatrix} K_1^{10} \\ K_{14}^{10} \\ K_{11}^{10} \\ K_8^{10} \end{bmatrix} \right) \quad (7) \end{aligned}$$

誤りを含む暗号文から秘密鍵を求める手順を次に示す。

- ① 10ラウンド目のラウンド鍵 K^{10} と ε を仮定し、 ε' を計算する。
- ② ε' および仮定した K^{10} 、実験で得た C^* 、計算で得た C を式(7)に代入し、矛盾なく定まる場合の K_i^{10} を候補鍵とする。
- ③ ①②を繰り返し、部分鍵の候補を求め、その中で最も多く現れる部分鍵を正解とする。

5. フォールト攻撃実験

SAKURA-GとSASEBO-GIIを用いて、まずAESの9ラ

ウンド目を対象としたグリッチによるフォールト攻撃実験を行った。諸条件を表1に示す。SAKURA-GおよびSASEBO-GII上で、制御用FPGAから暗号用FPGAに対するクロック信号は、FPGA間を基板上で直接接続するI/Oを使用している。

表1 実験の諸条件

	SAKURA-G	SASEBO-GII
FPGAのコア電圧	1.2V	1.0V
グリッチ挿入ラウンド	9ラウンド	
グリッチ幅[T]	10 (819.9ps)	10 (204.2ps)
グリッチ開始 [T]	45	35
グリッチ終了 [T]	55	45
入力平文	ランダム	
128bit AESの秘密鍵	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F	
暗号処理の回数	10,000	

グリッチを挿入しない通常の暗号処理の消費電力波形を図3に示す。クロックに同期して10ラウンドの暗号化処理の波形が観察される。また図4は9ラウンド目にグリッチを挿入して、誤った暗号文が出力された消費電力波形で、図3に対してピークの数が増加していることがわかる。これは9ラウンドの演算開始直後のグリッチの立ち上がりにより、10ラウンド目の処理が開始されたことによるものである。一方SASEBO-GIIは、シミュレーションでは正しくグリッチが入っていたが、実験では波形に変化が見られず、フォールトが生じなかった。



図3 フォールトなし AES 消費電力波形

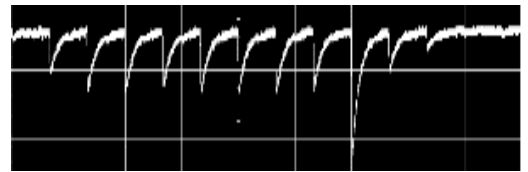


図4 フォールトあり AES 消費電力波形

暗号用FPGAはクロックの立ち上がりに同期している。フォールトに影響するのは図1の位相シフトクロック2の遅延によるグリッチ終了時刻（クロック1の遅延は開始時刻に影響）であるが、グリッチをきちんと入れるためにはある程度のグリッチ幅が必要である。そこでDCMで、シグニフィカント開始時刻のパラメータ T_e を誤りが発生しやすい65に固定し、グリッチ開始時刻 T_s を55~61の間で変化させることで、グリッチ幅 T_w を4~10としたときの、誤りを生じたバイト位置とその発生確率を10,000回暗号化を行い調べた。 $T_w=6$ 以上で100%の確率で誤りを生じている10バイト目と13バイト目において、グリッチ幅 $T_w=5$ では正常動作が15%ほどであった。オシロスコープで電力波形を観

測したところ、このとき正常時と同じ波形が見られ、 $T_w=4$ では誤りを生じないことから、フォールト誘発にはある程度のグリッチ幅が必要である。また終了位置 T_e が同じ場合、グリッチ幅 T_w によらず同一の誤りが生じることが確認されたため、以下では $T_w=10$ とし、グリッチ挿入のタイミングは、グリッチ終了時刻 T_e で制御することとする。

表2 グリッチ幅と誤り発生確率(%)

グリッチ開始 T_s	55	56	57	58	59	60	61
グリッチ終了 T_e	65	65	65	65	65	65	65
グリッチ幅 T_w	10	9	8	7	6	5	4
1 バイト目	0	0	0	0	0	0	0
2 バイト目	0	0	0	0	0	0	0
3 バイト目	0	0	0	0	0	1	0
4 バイト目	18	19	16	17	17	15	0
5 バイト目	0	0	0	0	0	0	0
6 バイト目	0	0	0	0	0	0	0
7 バイト目	2	2	1	1	1	3	0
8 バイト目	1	1	1	1	1	1	0
9 バイト目	1	1	1	1	1	1	0
10 バイト目	100	100	100	100	100	86	0
11 バイト目	0	0	0	0	0	0	0
12 バイト目	0	0	0	0	0	0	0
13 バイト目	100	100	100	100	100	84	0
14 バイト目	1	1	1	1	1	1	0
15 バイト目	0	0	0	0	0	0	0
16 バイト目	0	0	0	0	0	0	0

次にグリッチのタイミングを $T_e=45\sim 80$ の範囲で変化させ、固定の秘密鍵に対して 10,000 個のランダムな平文を暗号化しながら、9 ラウンド目の出力における各バイトの誤りの発生回数を調べた結果を図5に示す。またグリッチのタイミングを遅くしていくと、演算がグリッチの立ち上がりまでに間に合って誤りの数が減っていくことがわかる。13～16 バイト目の誤りが他に比べてなかなか減らないことから、ここに AES 回路のクリティカルパスがあることもわかる。なお同一のタイミングでも誤りが生じたり生じなかったりする主な原因は、平文によっても演算時間が異なるためである。逆に平文が同じであれば、同じグリッチのタイミングで同じ誤りが生じており、正確なフォールト制御が可能であることがわかる。

DFA は、9 ラウンド目の各 4 バイト列の入力に誤りが含まれていると仮定して解析を行うため、1 列中の誤りバイト数が多いほど、解析の計算量は増大する。1 列中に 1, 2, 3, 4 バイトの誤りが含まれている場合の数は、それぞれ ${}_4C_1 \times 255$ 通り、 ${}_4C_2 \times 255^2$ 通り、 ${}_4C_3 \times 255^3$ 通り、 ${}_4C_4 \times 255^4$ 通りで、さらに部分鍵の仮定が 256^4 あるので、4 バイト誤りでは $255^4 \times 256^4$ 通りの計算が必要となる。そこで各列に対して、1 バイトの誤りを起こせるかどうか解析において非常に重要である。

グリッチのタイミングは $T_e=60$ 以降では誤りが 13～16 バイト目に偏っている。これらの誤りは *ShiftRows* によって、それぞれ 4, 3, 2, 1 列目と全ての列に攪拌される。つまり、8 ラウンドの 13～16 バイト目に誤りを起こせれば、9 ラウンドで *ShiftRows* により各列 1 バイトのみ誤ったデー

タとなり *MixColumns* に入力される。表3にまとめた実験結果から、 $T_e=68\sim 76$ では 14 バイト目だけに誤りが生じ、 $T_e=67$ では 13 バイト目、 $T_e=66$ では 16 バイト目と増えてくことがわかる。したがって、本回路では 8 ラウンドの $T_e=60\sim 70$ が適したタイミングであることがわかる。

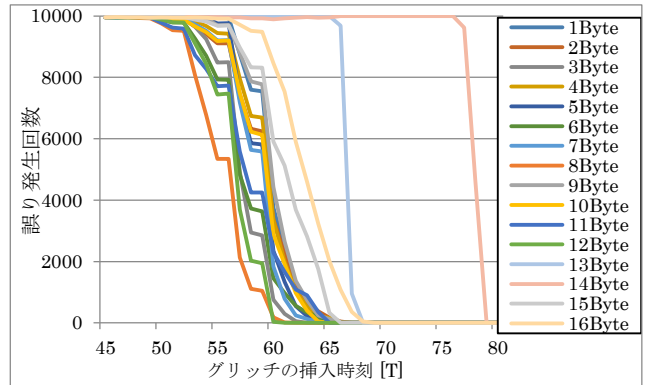


図5 グリッチの挿入時刻と9ラウンドでの誤り発生回数

表3 グリッチのタイミングと誤り発生確率

	バイト															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
35	99	99	99	99	99	99	99	99	99	99	99	99	99	99	99	99
36	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
37	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	99
38	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
39	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	99
40	100	100	100	100	100	100	100	100	99	100	100	100	100	100	100	100
41	100	99	100	100	100	100	100	100	100	100	100	100	99	100	100	100
42	100	100	100	100	100	100	100	100	100	100	100	100	100	99	100	100
43	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
44	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
45	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
46	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
47	100	100	100	99	100	100	100	100	100	100	100	100	99	100	100	100
48	100	100	100	99	100	99	100	99	100	100	100	100	99	100	100	100
49	100	100	99	99	100	99	100	99	100	100	100	100	100	100	100	99
50	100	99	99	99	100	99	100	98	100	99	98	99	100	100	100	100
51	100	99	99	99	100	98	99	95	100	99	96	98	100	100	100	99
52	100	99	99	99	100	98	99	95	100	99	96	98	100	100	100	99
53	100	97	96	98	100	93	97	81	100	97	87	91	100	100	99	100
54	100	94	92	97	100	87	95	68	99	95	83	84	100	100	98	99
55	100	91	85	94	98	79	92	53	97	92	77	74	100	100	97	99
56	100	91	85	94	98	79	92	53	97	92	77	75	100	100	97	99
57	88	75	49	80	73	48	72	21	88	75	56	37	100	100	90	97
58	76	63	29	67	59	37	56	11	79	62	43	20	100	99	83	95
59	75	62	29	67	58	36	56	11	78	61	42	19	100	99	83	95
60	38	34	8	34	24	15	18	2	44	29	23	0	100	99	59	85
61	24	23	3	20	13	10	8	0	27	19	17	0	100	99	51	75
62	11	12	1	10	5	6	2	0	14	10	11	0	100	99	37	59
63	6	7	0	5	2	3	1	0	7	4	9	0	100	100	28	46
64	2	4	0	2	0	2	0	0	2	1	4	0	100	99	18	32
65	0	2	0	2	0	0	0	0	1	0	0	0	100	100	3	20
66	0	0	0	0	0	0	0	0	0	0	0	0	97	100	0	11
67	0	0	0	0	0	0	0	0	0	0	0	0	10	100	0	4
68	0	0	0	0	0	0	0	0	0	0	0	0	0	100	0	0
69	0	0	0	0	0	0	0	0	0	0	0	0	0	100	0	0
70	0	0	0	0	0	0	0	0	0	0	0	0	0	100	0	0
71	0	0	0	0	0	0	0	0	0	0	0	0	0	100	0	0
72	0	0	0	0	0	0	0	0	0	0	0	0	0	100	0	0
73	0	0	0	0	0	0	0	0	0	0	0	0	0	100	0	0
74	0	0	0	0	0	0	0	0	0	0	0	0	0	100	0	0
75	0	0	0	0	0	0	0	0	0	0	0	0	0	100	0	0
76	0	0	0	0	0	0	0	0	0	0	0	0	0	100	0	0
77	0	0	0	0	0	0	0	0	0	0	0	0	0	96	0	0
78	0	0	0	0	0	0	0	0	0	0	0	0	0	46	0	0
79	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
80	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 4 誤りパターンの再現性実験の設定

	実験 1	実験 3	実験 2	実験 4
終了 T_e	50	60	50	60
平文	平文 1~10			
秘密鍵	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 3F 1C 77 C5 A8 6E 5A F1 19 A4 07 3F 51 FD AE A7			

表 5 平文入力

平文 1	00 11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF
平文 2	63 85 47 81 88 0E 47 7E D8 9C 1A BE 66 EE 21 B3
平文 3	B9 D1 C4 8E 34 8F E7 71 FA 46 4A 77 A1 78 FB 07
平文 4	DC FE AD 50 D1 D9 FD 08 B3 86 EF B0 8B 14 2F 74
平文 5	4C FD A4 C3 07 61 57 F5 81 B7 1E 46 E8 CB 56 75
平文 6	25 38 68 B7 9E 7C 31 E0 D3 BC DB AE 9F 37 E5 E5
平文 7	16 D6 2B A4 D3 38 FF 7B A6 B8 CF 19 BA B7 20 E7
平文 8	30 7A 8A 53 AB 77 D5 CF CB C3 F0 4B 85 A9 55 49
平文 9	D2 38 2F 11 9F B1 A1 ED 38 ED 11 E6 AD 18 26 1B
平文 10	9C F3 7D 25 AD 91 39 9E 01 5C D9 0F 48 B4 F5 FD

表 6 誤りパターンの再現性実験結果 ($T_e=50$)

		同一誤りを含む暗号文の個数				
		1 位	2 位	3 位	4 位	5 位
実験 1	平文 1	1,406	1,186	987	809	783
	平文 2	3,868	1,728	469	465	421
	平文 3	899	774	739	660	660
	平文 4	2,801	1,242	1,227	768	584
	平文 5	2,954	1,518	668	658	583
	平文 6	2,748	1,096	809	621	490
	平文 7	1,706	930	802	794	720
	平文 8	1,676	1,124	587	498	498
	平文 9	2,794	1,749	765	658	464
	平文 10	1,145	985	735	735	722
実験 2	平文 1	1,448	1,100	1,018	663	504
	平文 2	3,260	1,577	1,380	522	496
	平文 3	1,360	921	639	572	503
	平文 4	1,383	1,348	1,048	869	549
	平文 5	611	475	360	350	276
	平文 6	1,952	942	426	340	285
	平文 7	1,053	1,008	696	685	467
	平文 8	733	718	641	508	500
	平文 9	1,488	985	826	598	475
	平文 10	703	538	524	373	365

表 7 誤りパターンの再現性実験結果 ($T_e=60$)

		同一誤りを含む暗号文の個数				
		1 位	2 位	3 位	4 位	5 位
実験 3	平文 1	9,228	660	70	27	12
	平文 2	10,000	なし			
	平文 3	6,100	3622	154	124	なし
	平文 4	3,840	2,070	1,643	538	486
	平文 5	5,383	4,608	7	2	なし
	平文 6	9,996	3	1	なし	
	平文 7	3,263	2,084	1,765	1,494	1,174
	平文 8	4,354	4,337	948	230	111
	平文 9	8,837	500	218	172	125
	平文 10	6,835	2,929	144	48	35
実験 4	平文 1	5,708	3,132	600	543	16
	平文 2	5,446	1,734	878	862	817
	平文 3	7,655	2,004	285	56	なし
	平文 4	6,216	1,217	740	649	332
	平文 5	9,968	24	4	2	
	平文 6	6,967	3,028	5	なし	
	平文 7	7,634	1,862	470	33	1
	平文 8	8,664	1,336	なし		
	平文 9	7,031	1,999	672	176	122
	平文 10	5,419	3,616	525	320	119

次に誤りパターンに再現性があるかどうかの確認実験を行った。表 4 の設定と表 5 の 10 通りの平文に対して、それぞれ 10,000 回の暗号処理中にフォールトを起こし、同一の誤りが発生する回数を調べた。表 6 と 7 に、各実験（実

験 1~4×平文 1~10）における誤りを含む暗号文の出現回数
の 1~5 位を示す。 $T_e=60$ では 9 割以上で同じ暗号文が出現しているのに対し、 $T_e=50$ では 6 割程度にとどまっている。平文によって演算時間が異なるので、微妙なタイミングによっては誤りパターンが異なってくるが、これらの実験から非常に高い再現性を有していることがわかる。

グリッチのタイミングを $T_e=60\sim70$ として 8 ラウンドにフォールトを起こすことで、DFA 解析に有用な 1 バイト誤りを各列に起せることは図 6 から明らかであるが、実際にその通り DFA が可能かどうかを表 8 の条件で検証した。その結果が表 9 であり、導出できた部分鍵を“1”でできなかったものを“0”で示している。結果 $T_e=62,64,65$ のときにすべての部分鍵を導出することに成功している。また、調べた範囲すべてで部分鍵が導出されているグループ C に比べ、グループ B のように部分鍵が導出しづらいものもあった。このことから高い精度でのフォールト混入タイミングの決定が必要であることが分かる。

表 8 DFA 実験条件

	SAKURA-G
FPGA のコア電圧	1.2V
グリッチ挿入ラウンド	8
グリッチ幅 T_w	10 (819.9ps)
グリッチ終了 T_e	60~71
平文	ランダム
秘密鍵	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
暗号処理の回数	50,000

表 9 DFA 結果

部分鍵	13	11	1D	7F	E3	94	4A	17	F3	7	A7	8B	4D	2B	30	C5
グループ	A	B	C	D	B	C	D	A	C	D	A	B	D	A	B	C
グリッチ挿入時刻 T_e	71	0	0	1	0	0	1	0	0	1	0	0	0	0	0	1
	70	0	0	1	0	0	1	0	0	1	0	0	0	0	0	1
	69	0	0	1	0	0	1	0	0	1	0	0	0	0	0	1
	68	1	0	1	0	0	1	0	1	1	0	1	0	0	1	0
	67	1	0	1	1	0	1	1	1	1	1	0	1	1	0	1
	66	1	0	1	1	0	1	1	1	1	1	0	1	1	1	1
	65	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	64	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	63	1	0	1	1	0	1	1	1	1	1	0	1	1	1	1
	62	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	61	1	0	1	0	0	1	0	1	1	0	1	1	0	1	0
	60	0	0	1	0	0	1	0	0	0	0	0	0	1	0	1

6. むすび

本論文では、SAKURA-G および SASEBO-GII 上に DCM および PLL を用いたグリッチ回路を実装し、AES 暗号回路に対して DFA によるフォールト攻撃実験を行った。SASEBO-GII はメインの暗号 FPGA の Viretx-V に対して制御用 FPGA が低速な Spartan-3 であったため、高速かつ正確なタイミングでグリッチを発生することができず、誤動作を生じなかった。その一方、暗号用と制御用に同じ Spartan-6 を用いた SAKURA-G は、高い精度と再現性で DFA が可能な誤りを誘発することに成功した。

グリッチの挿入位置を細かくずらしながら、発生するバ

イト誤りを調べる実験では、演算が遅いクリティカルパスを同定し、そこにタイミングを合わせることで DFA に適した少ないバイト数の誤りを発生させられることを示した。

DFA では誤ったバイト数が多くなるにつれ、解析のための演算量が急激に増大するが、これは可能な鍵や誤りパターンの組み合わせが増えるためである。暗号回路に誤動作を起せたならば、誤動作のタイミングを制御しながら、平文、暗号文そして秘密鍵を与えて DFA を実施することができる。この場合、既知の秘密鍵を用いているので多数の組み合わせを試すのではなく、所望の誤りを生成することができるかどうかを調べることになる。これによって、秘密鍵を知らずに DFA を行う場合に比べ、少ない演算量でより正確な脆弱性評価を行うことが可能となる。このように内部情報にアクセスできることを前提としたホワイトボックス評価について、今後検討を進めていきたい。

参考文献

- [1] D. Boneh, “On the importance of checking cryptographic protocols for fault,” EUROCRYPT ’97, Vol 1233, pp.37-51, 1996
- [2] P. Kocher, “Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems,” CRYPTO ’96, LNCS 1109, pp. 104-113, Aug. 1996.
- [3] AIST, “Evaluation Environment for Side-Channel Attacks.” <http://www.risec.aist.go.jp/project/sasebo/>
- [4] “SAKURA hardware security project” <http://www.morita-tech.co.jp/SAKURA/en/index.html>
- [5] Xilinx, “Spartan-6 FPGA クロックリソース” UG382 (v1.3), 2010 年
- [6] NIST, “Advanced Encryption Standard (AES),” FIPS-197, Nov. 2001. <http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf>
- [7] A. Moradi, et al., “A Generalized Method of Differential Fault Attack AES” CHES 2006, LNCS 4249, pp. 91-100, Oct. 2006.