Security Evaluation of Different AES Implementations Against Practical Setup Time Violation Attacks in FPGAs.

Conference Paper · January 2009

DOI: 10.1109/HST.2009.5225057 · Source: DBLP

CITATIONS
15

READS
106

4 authors, including:

Shivam Bhasin
MINES ParisTech
128 PUBLICATIONS **996** CITATIONS
SEE PROFILE

Jean-Luc Danger
Institut Mines-Télécom
241 PUBLICATIONS **2,566** CITATIONS
SEE PROFILE

Sylvain Guilley
Institut Mines-Télécom
297 PUBLICATIONS **3,163** CITATIONS
SEE PROFILE

Security Evaluation of Different AES Implementations Against Practical Setup Time Violation Attacks in FPGAs

Shivam BHASIN Nidhal SELMANE Sylvain GUILLEY Jean-Luc DANGER
Institut TELECOM, TELECOM ParisTech, CNRS LTCI (UMR 5141) — TCP Project
Departement COMELEC, 46 rue Barrault
75 634 PARIS Cedex 13, FRANCE
Email: < firstname.surname@telecom-paristech.fr >

***Abstract*— Security evaluation of various AES implementation against practical power attacks has been reported in literature. However, to the authors' knowledge, very few of the fault attacks reported on AES have been practically realized. Since sbox is a crucial element in AES, in this article, we evaluate the security of some unprotected AES implementations differing in sbox construction, targeted for FPGA. Here the faults have been generated practically by underpowering the targeted circuit. Then we correlate our results with the underlying architecture, along a methodology already suggested in other articles, albeit theoretically. We also carry out an extensive characterization of the faults, in terms of temporal localization. On the basis of our results, we reach the conclusion that the two cheaper implementations in terms of silicon area are also the more vulnerable against DFA when implemented without countermeasures.**

I. Introduction

Side channel attacks (SCA) target directly the physical implementation of a cryptographic system in order to retrieve its secret key. These attacks can be classified into two types, both of which provide enough information to fully compromise the security. Passive attack which comprises of observing the physical emanations of the system, like power (Differential Power Analysis, or DPA [1]) or E/H field (ElectroMagnetic Analysis, or EMA [2]), are two representative kinds of SCA. In such attacks, an off-line analysis of the physical measurements allows to extract the full key, either by correlation [3] or by pattern matching [4] techniques.

The second kind of SCA is known as active attack. It involves the injection of faults during the execution of a cryptographic algorithm. Malicious techniques based upon the variations of supply voltage, clock frequency, temperature variation, or irradiation by a laser beam will most probably lead to a wrong computation result that can be exploited. Such attacks will enable an attacker to retrieve secret information concealed within the device [5]. From the knowledge of one or multiple couples {correct ciphertext, faulted ciphertext}, some hypotheses on the secret key can be discarded. In this respect, DFA is a special case of the "impossible differential cryptanalysis" [6] attack framework. This generic attack strategy is referred to as DFA (Differential Fault Analysis [7]). Although active attacks were reported later (in 2001 [8]) than passive attacks (in 1998 [1]), it is shown in the literature that this method requires fewer interactions with the device as compared to passive attack. This kind of attack represents a real threat for the implementation of cryptographic algorithms such as the advanced encryption standard (AES).

The winning AES block cipher algorithm was published by the NIST in 2001 [9]. AES can operate on a message of 128 bit with three different key sizes: 128, 192 and 256 bit. In the sequel, and without loss of generality, we focus on the 128-bit version of AES. The algorithm AES is a substitution permutation network (SPN) product block cipher. It has an iterative structure, consisting of the serial repetition of ten identical rounds which is applied to the 16 bytes message block to be encrypted. The 16 bytes are laid out as a matrix of four columns of four bytes $s_{i,j}$, where $0 \leq i \leq 3$ and $0 \leq j \leq 3$. A round consists of a fixed sequence of transformations. Apart from the first and the last rounds, the other nine rounds are alike and consist of four transformations each. The first and last rounds are incomplete to ease the decryption. The four round transformations are called SubBytes, ShiftRows, MixColumns and AddRoundKey.

When considering hardware implementation, ShiftRows is a simple swapping of wires. MixColumns can be implemented with shift operations & XOR gates. AddRoundKey consists of XOR operation only. SubBytes, the non-linear part should be implemented with special considerations as it provides major strength to the algorithm. This can be done in two different ways. First, the substitution boxes (in short "sbox", the AES SubBytes combinatorial function) can be implemented as a table as described in the standard [9]. The other method to implement the sbox could be calculation of the values at run-time performing multiplicative inverse in GF(2^8) & affine transformation in GF(2). Since the SubBytes is also the most bulky part of AES, area used by sbox is important for hardware implementations. If we decide to implement the cryptographic processor as hardware instead of software, it is to make the implementation as efficient as possible. Otherwise it is not worth the effort.

Since the architecture of sboxes are different the propagation

978-1-4244-4804-3/09/$25.00 © 2009 IEEE

delay is also different, so will be the affect of faults on such architectures. When the sbox are implemented as a table the input to the SubBytes serves as an address to the table. When a fault occurs, it changes the address and thus the output byte. This change in address does not takes a lot of time to propagate. On the other hand, operation in Galois field suffers a reasonable propagation delay. This delay will be favourable for fault attacks. This is further confirmed in our experimental results.

We have designed an AES co-processor targeted for FPGA. Owing to the above mentioned methodology three different types of substitution boxes are tested. One of the three sbox is implemented as a table in LUT (lookup tables or FPGA logic blocks). To make the design area efficient, we have moved the sbox to RAM in the second implementation. In the third design we use the sbox in binary Galois field GF(2^4) (a variant of the latter calculation method) as described in [10]. The first & the third sbox are similar to what have been discussed in [11] as LUT & GF(2^4). Also the MixColumns used is similar in construction to what has been called as "conditional addition MixColumns" in [11].

In [11], the authors have compared various AES implementations on the basis of simulated timing analysis of a delay fault model. These implementations differ from each other in either constructin of SubBytes or in MixColumns. Authors suggest a term "Attack Frame" which describes the precision required on the delay introduced to have an exploitable fault in the design. If the delay is not in the range of attack frame, this may lead to no fault or multiple faults. On the basis of attack frame, authors generate a table to compare various attack frames which corresponds to security of the design.

The results presented in this article are obtained with an EP1S25 Altera FPGA soldered in a Parallax evaluation board. As described in [12], [13], faults can practically be induced on an FPGA by underpowering the circuit. When we drive the FPGA at a voltage less than nominal voltage, the propagation time of the signal increases as illustrated in Figure 1. Such attacks are non-invasive in nature as the attacker does not need access to the silicon die and therefore are easier to implement. We recall that there is no straightforward mechanism to monitor either the power supply level or the frequency in commodity FPGAs. This phenomenon causes a setup time violation on one of the timing path of the design causing a faulty byte. We call this fault as a byte-flip fault caused by flipping of one or more bits in a byte which can be observed by monitoring the hamming weight. Hamming weight of a byte can be defined as the number of non-zero bits in a byte. Since cryptography involves highly complex computations it is very likely that the critical path is in the cryptographic part [14]. Such faults can be exploited using various known attacks [15], [16], [17]. Here we use the Piret's attack to exploit the faults and retrieve the secret key using the method as described in [15].

In this article we characterize the architectural features that makes such attacks more or less successful. In this respect, we try to evaluate security of three implementations against setup violation faults. Later in the text, we make an attempt to relate our results to those published in [11].

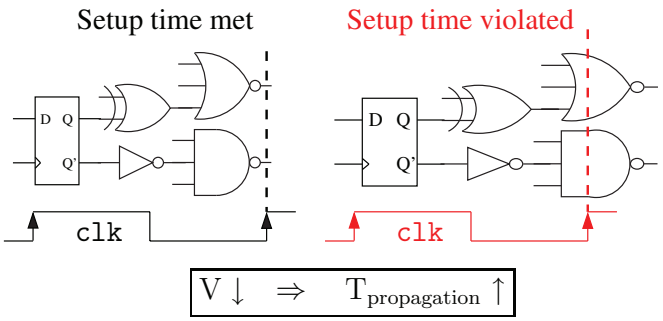

Fig. 1. Setup time violation caused by a permanent under-voltage.

The rest of the paper is organized as follows. In section II, the architecture of AES co-processor is discussed briefly. In section III, we explain our experimental setup. section IV presents the results from a fault & power acquisition campaign on various versions of AES and their comparative study in terms of spatial and temporal localization. Finally, section V concludes the paper and opens perspectives for better protecting sensitive cryptographic implementation.

II. AES Architecture

Figure 2 shows the architecture of a simple, non protected AES co-processor. The AES co-processor is designed to have a parallel architecture. It performs each round of AES in each clock cycle. The four sub-rounds are SubBytes, ShiftRows, MixColumns and AddRoundKey. These sub-rounds along with some multiplexers and key scheduler comprise the datapath. The key scheduler or expander calculates a key for each round which is then used in the datapath.

The SubBytes & Key Schedule use 16 & 4 sboxes respectively. However, the design of sbox is different in each design. In the first and second implementation, sbox is implemented as a table using "case/select" statement. This table is asynchronous and synthesized in LUTs for the former implementation. The latter implementation also uses the same design but the table is made synchronous and is sensitive to falling edge of the clock. Such implementations are automatically moved to the block RAM by the synthesis tool. At each falling edge, RAM samples the input address. The third implementation is based on finite field arithmetic instead of look up tables. As described by authors in [10], sbox operation in GF(2^8) can be implemented only with combinatorial logic. The operations in GF(2^8) can be done in GF(2^4) by representing the original polynomial as a linear polynomial with coefficients of four bits each. This sbox is implemented completely in LUTs.

III. Experimental Setup

The co-processor along with a UART interface and a controller are synthesized on the FPGA as shown in figure 3. This design communicates with a monitoring PC via RS-232 cable. Figure 4 sketches the experimental setup. The power

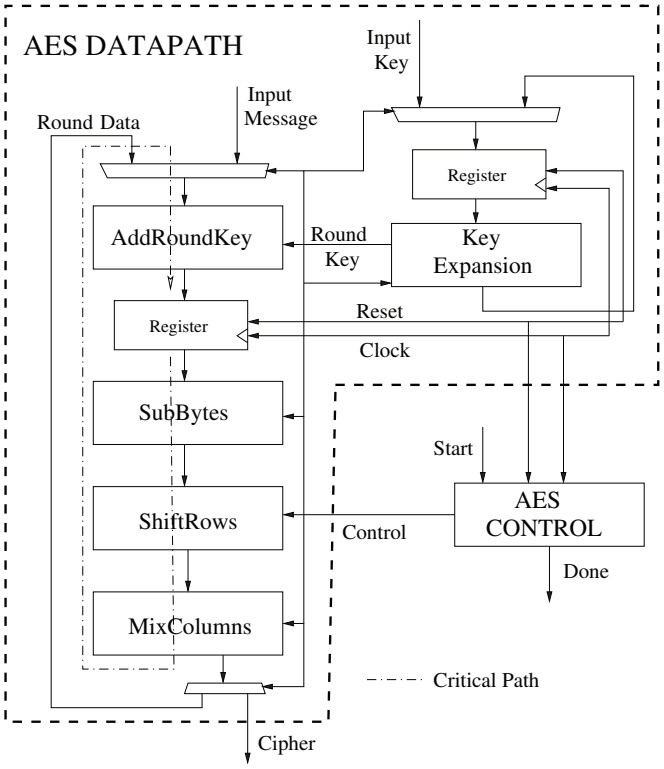

Fig. 2. AES architecture.

supply is controlled remotely, in order to test a set of non nominal values of Vcc successively.

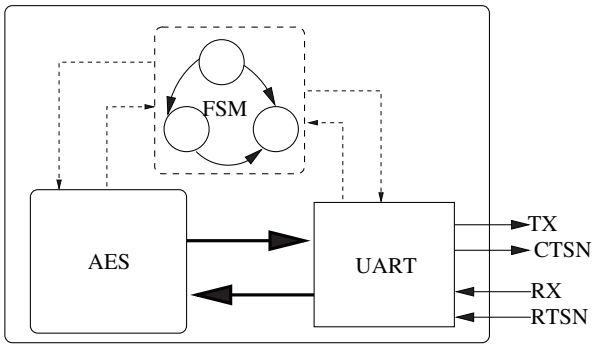

Fig. 3. The whole design synthesized on the FPGA.

The nominal voltage of the FPGA running the co-processor is 1.5 volts. The operating frequency is 50 MHz. We observe that FPGA remains functional for lower Vcc until eventually the module starts giving erroneous results. In order to collect faulted ciphertext for each architecture we have recorded the triples {message, key, ciphertext} for 1,000 encryptions at each 100 values of Vcc. As a result, the entire acquisition campaign consists of 100,000 encryptions for each architecture. After collecting this set triples, we analyze the fault as explained in article [12].

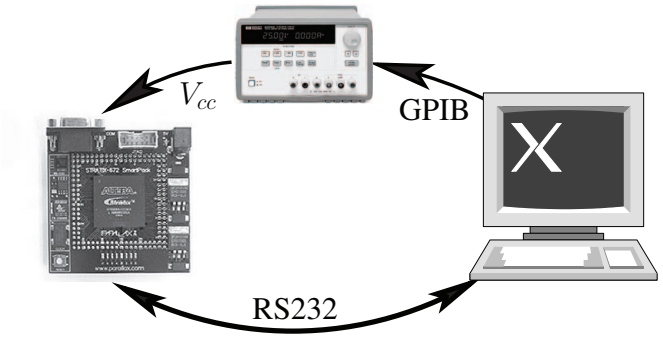

Fig. 4. Experimental attack platform.

IV. Experimental Results

In our architecture, the delays in the datapath are greater than in the key schedule. As we focus on non-invasive attacks, we assume local faults cannot be injected directly into the keypath. Moreover, as global perturbations will not affect the keypath either, we assume that the key schedule block is fault-free. At higher voltages (*i.e.* close to the nominal voltage) only single faults occur. As we decrease the voltage beyond a certain threshold, setup time is violated on multiple paths and faults become multiple (uncovered). It is straightforward to adapt the results obtained in this section to other attacks, such as attacks on the key schedule [18], [19].

Figures 5, 6, 7 show the occurrence of faults in the three architectures. Faults are partitioned into single *i.e.* faults which affect one byte on the AES state before the SubBytes transformation in the datapath or multiple *i.e.* faults affecting multiple byte or occur in the keypath. Single faults have a "bell-shape" distribution. This behavior is compatible with a fault model where errors are caused by a setup violation on critical combinatorial path. It is well established that propagation time of a signal on a particular path increases with decreasing supply voltage. Thus at lower voltages it is more likely that a critical path is violated to generate frequent single faults. Nevertheless, below a threshold, multiple critical paths are violated.hence an augmentation of multiple faults, and a subsequent diminution of single faults.

Figures 8, 9, 10 present the coverage of single faults, *i.e.* the ratio between single and detected faults. The first faults (for the higher voltage values), are almost all single as the coverage is close to one hundred percent. As the voltage decreases, the coverage degrades, attesting the gradual appearance of multiple faults. In our experiments, we use the "Piret's Attack" [15] to exploit the faults. As per this attack, single faults affecting only the two penultimate rounds are used for retrieving the key. From here, we address such faults as exploitable faults.

Figures 11, 12 and 13 show the Hamming weights of the exploitable byte-flips. One interesting observation from these figures is that most of the faults occurring in the circuit are a single bit fault (Hamming Weight of the fault=1). This information allows attacker to mount some of the published

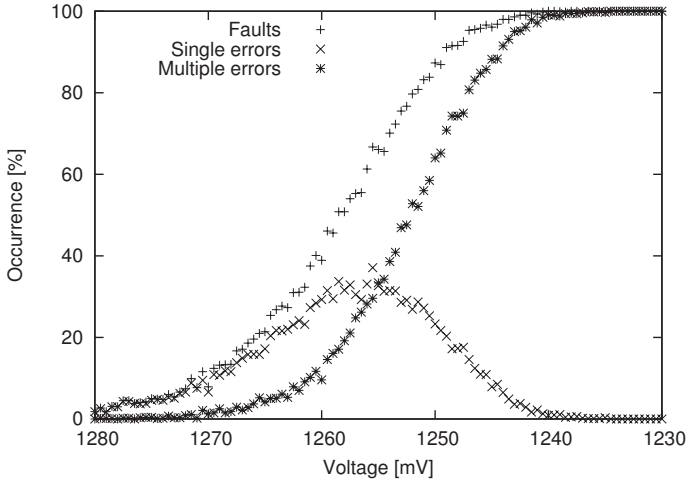

Fig. 5. Occurrence of faults: sbox in $GF(2^4)$.

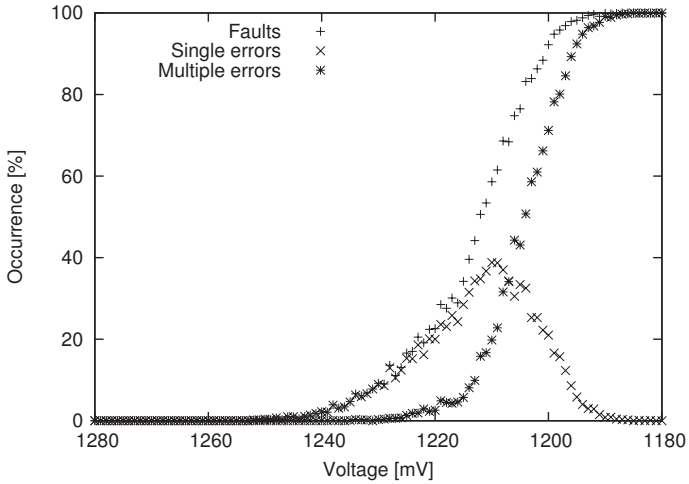

Fig. 6. Occurrence of faults: sbox in LUT.

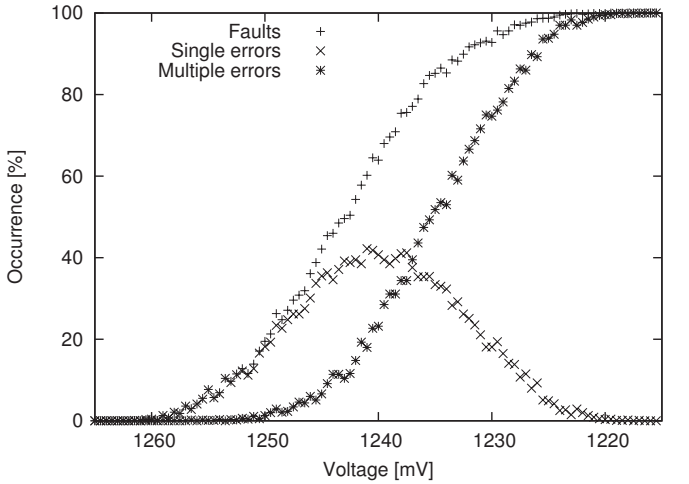

Fig. 7. Occurrence of faults: sbox in RAM.

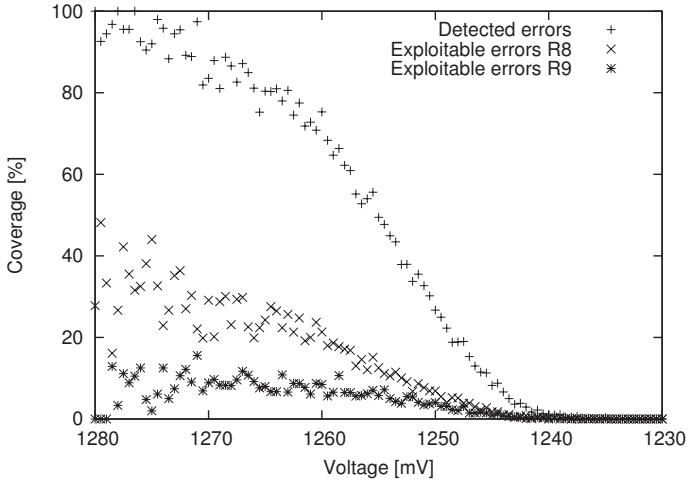

Fig. 8. Coverage of single faults, and detail of exploitable faults in $GF(2^4)$.

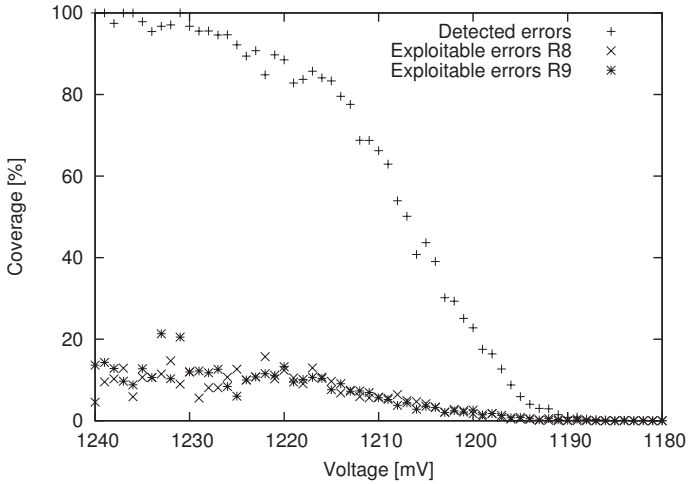

Fig. 9. Coverage of single faults, and detail of exploitable faults in LUT.

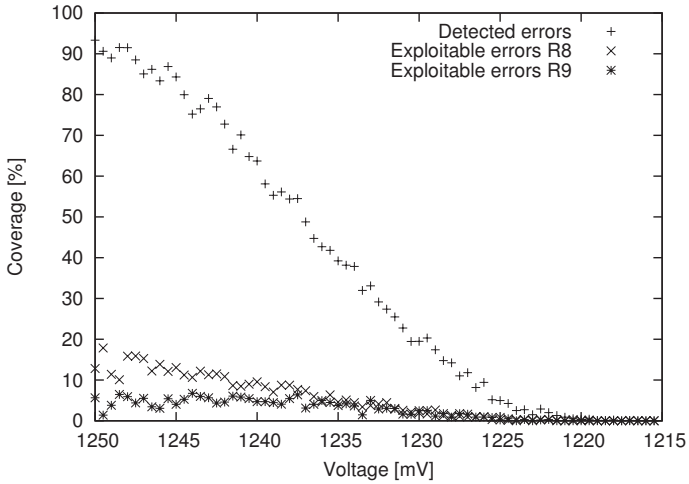

Fig. 10. Coverage of single faults, and detail of exploitable faults in RAM.

"Bit Fault" attacks [18].

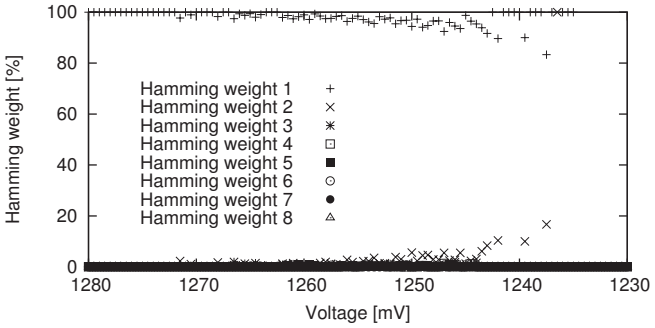

Fig. 11. Hamming weight of exploitable faults in $GF(2^4)$.

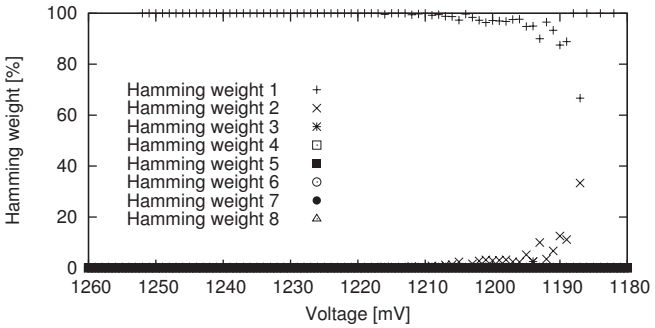

Fig. 12. Hamming weight of exploitable faults in LUT.

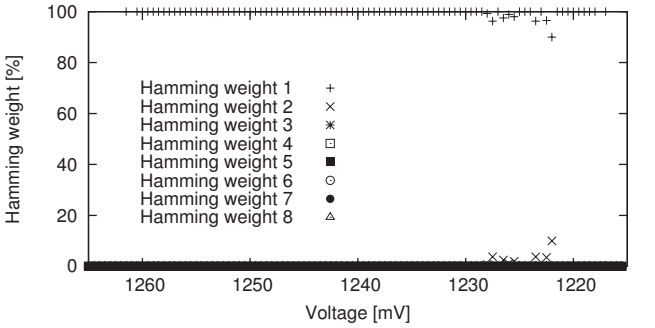

Fig. 13. Hamming weight of exploitable faults in RAM.

Figures 14 and 15 show the temporal and spatial localization of the single faults. In figure 14 there is no fault in first round. This is because first round in AES is comprised only of AddRoundKey operation resulting in a fairly small timing path. This also proves that the communication between the FPGA and PC is fault free. In figure 15, each sbox has different number of faults. Since the computation time of logic gates is data-dependent, there is an uneven temporal and spatial distribution of the faults.

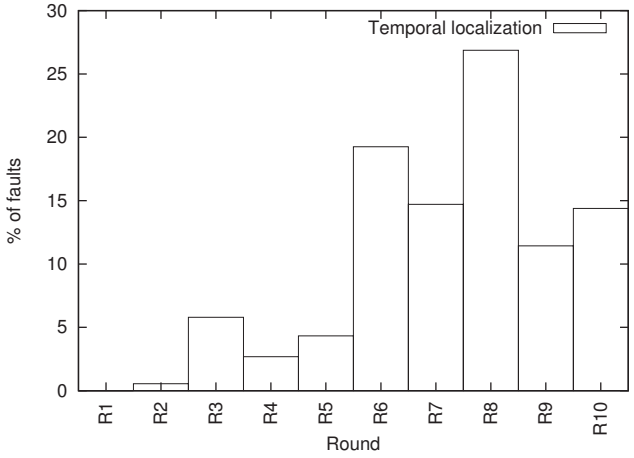

Fig. 14. Temporal localization of single faults.

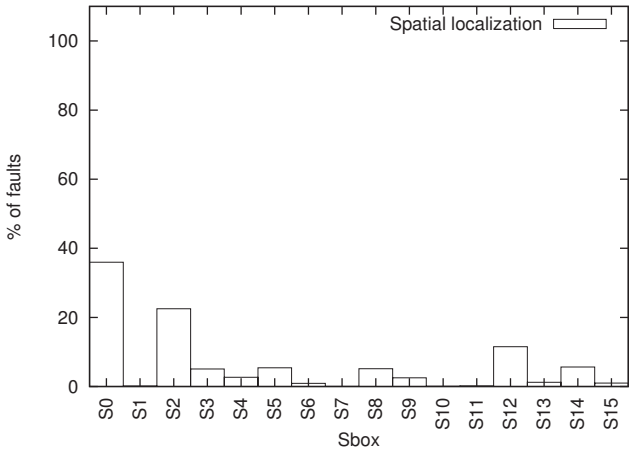

Fig. 15. Spatial localization of single faults.

A. Cost Comparison of the Three Architectures

For the purpose of cost comparison (in terms of area), we synthesized the AES co-processor with three different sboxes using Altera Quartus. The results are summarized in the table I. Here we see that AES co-processor with sbox (table) in LUT uses maximum area while the one with sbox in RAM uses minimum area as sbox is the most bulky part. This further multiplies because of the fact that we use multiple instances of sbox in this parallel architecture. When the sbox is synthesized using GF(2^4), each sbox takes almost 4 times lesser area than the one in LUT. Hence we reduce a lot of cost in terms of area. Every architecture uses 256-bit registers as it memorize the round key and round data once per clock.

B. Security Evaluation of the Three Architectures against DFA

In this section, we compare the three architectures with respect to security. Figures 5, 6, 7 show the occurrences of faults in different architectures. For the sake of comparison, we plot the exploitable faults on the same diagram. In figure 16 we see that the peak of the bell shaped distribution is highest

Architecture	LUTs	LUTs/Sbox	Critical Path in Datapath
LUT	5212 (20%)	206	13.725 ns
RAM	1061 (4%)	0	19.818 ns
GF(2^4)	2280 (9%)	60	17.569 ns

TABLE I

COST COMPARISON OF THE THREE STUDIED ARCHITECTURES.

for the architecture with sbox in GF(2^4). It shows that around 13% of the single faults are exploitable. On the other hand, less than 6% of the faults are exploitable when the sbox is implemented as a table in LUT. This means that we need half the amount of faulty ciphertext when attacking sbox in GF(2^4) than needed for LUT. These results are in accordance with the results obtained in [11], where authors have used the timing analysis of post map netlist of AES co-processor. Authors demonstrate that it is more difficult to attack sbox in LUT than a sbox in GF(2^4) because a higher attack frame means higher probability of creating a single fault. Thus, we have prqcticqlly proven the results which were stated theoretically in [11]. Voltage is another parameter for security. A sbox which gets faulty at lower voltage is more secure because it is more likely that some other part of the design stops working at lower voltages.

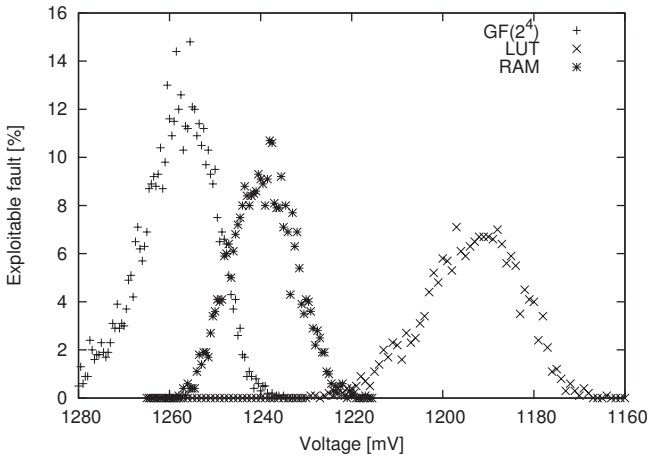

Fig. 16. Exploitable errors.

Recently, few methods have been reported [20] which suggest to synthesize the bulky parts of AES like SubBytes & MixColumns into the peripherals like block RAM, DSPs etc. These methods reduce the logic utilization in the FPGA and hence are cost effective. We also tested an sbox in the RAM. The results as shown in figure 16, 9% of the faulty ciphertexts are exploitable as compared to 6% in case of sbox in LUT. So we see there is a trade-off between cost & security. It has always been known that higher security comes at higher cost; this rule of thumb also applies to AES. It is upto the designer to make an intelligent choice.

Figure 16 shows that sbox in RAM is more secure than the sbox in GF(2^4). On the other hand in table I, the timing information on critical path in the datapath suggest that sbox in RAM should be less secure than sbox in GF(2^4). Contrary results can be explained by following arguments. The clock period is 20 ns. The RAM is sampled on the falling edge while the state register is sampled at the rising edge. The critical path calculated is between these two edges which is just for negative half of the clock cycle. Quartus normalizes the timing depending on the duty cycle of the clock and displays in terms of one full cycle. This fact was confirmed when reduction in the duty cycle of the clock resulted in a higher maximal frequency of operation. In practice, the timing of the path is less than 19.818 and should be interpreted separately for each half of the clock cycle. This is also corroborated by figure 16. In case of sbox in GF(2^4), timing information given by quartus is trustworthy as all the operation are sensitive to rising edge of the clock. When the sbox is implemented in GF(2^4), as shown in the architecture, the worst-case critical path in the datapath will begin from and end at the state register which stores round data. In case of sbox in RAM, the worst-case critical path is between the output of RAM & state register. As compared to RAM, apart from operators of ShiftRows, MixColumns & AddRoundKey, the sbox in GF(2^4) also uses combinatorial operators to implement the sbox as well. Since occurrence of fault is a dynamic parameter, the presence of larger number of combinatorial components may increase the probability of occurrence fault in this architecture. However nothing definite can be concluded as the construction of RAM and LUTs are different. Due to the difference in construction, the delays due to underpowering will evolve differently.

V. CONCLUSION & FUTURE WORK

In this article, we have presented an evaluation of security of various AES implementations against setup time violation attacks. As compared to other architectures we found that sbox in LUT proves to be most robust implementation. We make an attempt to relate our scenario with what has been published elsewhere [11], [20]. In terms of cost, the sbox in LUT consumes the maximum area followed by sbox inGF(2^4) and RAM respectively. Sbox in GF(2^4) is good choice in terms of cost if the whole design has to be implemented in LUTs. Hence we conclude that the two cheaper implementations in terms of silicon area are also the more vulnerable against DFA when implemented without counter-measures. In case a different FPGA is used, though the critical path of each implementation might change, still the results should be propotional as the critical path depends mostly on the logic synthesized by the FPGA.

Our further task is to secure these unprotected AES implementations with some known countermeasures. Wave Dynamic Differential Logic (WDDL) [21] is one of the well known countermeasures. Many publication provide results of DPA against WDDL [22] but none to authors knowledge evaluate affect of faults on WDDL. We wish to evaluate the effects of fault attacks on such countermeasures.

REFERENCES

[1] P. Kocher, J. Jaffe, and B. Jun, “”Differential Power Analysis””, in *CHES99, LNCS Springer*, 1999, vol. 1666, pp. 388–397.

[2] J.-J. Quisquater and D. Samyde, “”ElectroMagnetic Analysis (EMA) Measures and Counter-Measures for Smart Cards””, in *e-SMART, LNCS Springer*, 2001, vol. 140, pp. 200–210.

[3] ric Brier, Christophe Clavier, and Francis Olivier, “Correlation Power Analysis with a Leakage Model”, in *CHES*. August 11–13 2004, vol. 3156 of *LNCS*, pp. 16–29, Springer, Cambridge, MA, USA.

[4] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi, “Template Attacks”, in *CHES*. August 2002, vol. 2523 of *LNCS*, pp. 13–28, Springer.

[5] Johannes Blmer and Jean-Pierre Seifert, “Fault based cryptanalysis of the Advanced Encryption Standard”, in *Financial Cryptography*, Springer, Ed., 2003, vol. 2742 of *LNCS*, pp. 162–181.

[6] Lars Knudsen, “DEAL – A 128-bit Block Cipher”, February 21 1998.

[7] Eli Biham and Adi Shamir, “Differential Fault Analysis of Secret Key Cryptosystems”, in *CRYPTO*. 1997, vol. 1294 of *LNCS*, pp. 513–525, Springer.

[8] Dan Boneh, Richard A. DeMillo, and Richard J. Lipton, “On the Importance of Eliminating Errors in Cryptographic Computations”, *Journal of Cryptology*, vol. 14, no. 2, pp. 101–119, 2001.

[9] Federal Information Processing Standards (FIPS) Publication, *Announcing the Advanced Encryption Standard (AES)*, Number 197. November 2001.

[10] Johannes Wolkerstorfer, Elisabeth Oswald, and Mario Lamberger, “An ASIC Implementation of the AES SBoxes”, in *CT-RSA*, Bart Preneel, Ed. 2002, vol. 2271 of *Lecture Notes in Computer Science*, pp. 67–78, Springer.

[11] Julien Francq and Olivier Faurax, “Security of several AES Implementations against Delay Faults”, in *Proceedings of the 12th Nordic Workshop on Secure IT Systems (NordSec 2007)*, October 2007, Reykjavk, Iceland.

[12] Nidhal Selmane, Sylvain Guilley, and Jean-Luc Danger, “Setup Time Violation Attacks on AES”, in *EDCC, The seventh European Dependable Computing Conference*, Kaunas, Lithuania, may 2008, pp. 91–96, ISBN: 978-0-7695-3138-0, DOI: 10.1109/EDCC-7.2008.11.

[13] Farouk Khelil, Mohamed Hamdi, Sylvain Guilley, Jean-Luc Danger, and Nidhal Selmane, “Fault Attack on AES FPGA Encryption Platform”, in *NTMS*, Tangier, Morocco, nov 2008, pp. 1–5, IEEE.

[14] Olivier Faurax, Assia Tria, Laurent Freund, and Frederic Bancel, “Robustness of circuits under delay-induced faults : test of aes with the pafi tool”, *IEEE International On-Line Testing Symposium/On-Line Testing Symposium, IEEE International*, vol. 0, pp. 185–186, 2007.

[15] Gilles Piret and Jean-Jacques Quisquater, “A Differential Fault Attack Technique against SPN Structures, with Application to the AES and KHAZAD”, in *CHES*. September 2003, vol. 2779 of *LNCS*, pp. 77–88, Springer, Cologne, Germany.

[16] Chien-Ning Chen and Sung-Ming Yen, “Differential fault analysis on AES key schedule and some countermeasures”, in *Information Security and Privacy*, Springer, Ed., 2003, vol. 2727 of *LNCS*, pp. 118–129.

[17] Hagai Bar-El, Hamid Choukri, David Naccache, Michael Tunstal, and Claire Whelan, “The Sorcerer’s Apprentice Guide to Fault Attacks”, *Proceedings of the IEEE*, vol. 94, no. 2, pp. 370–382, 2006, DOI: 10.1109/JPROC.2005.862424.

[18] Christophe Giraud, “DFA on AES”, in *Advanced Encryption Standard (AES) 4th international conference, LNCS springer*, Springer, Ed., May 2005, vol. 3373 of *LNCS*, pp. 27–41, Bonn, Germany.

[19] Chong Hee Kim and Jean-Jacques Quisquater, “New Differential Fault Analysis on AES Key Schedule: Two Faults are enough”, in *CARDIS 2008*. 2008, Springer, Royal Holloway, University of London, UK.

[20] Saar Drimer, Tim Gneysu, and Christof Paar, “DSPs, BRAMs and a Pinch of Logic: New Recipes for the AES on FPGAs”, in *IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM)*. 14-15 Apr 2008, pp. 99–108, IEEE, Stanford, Palo Alto, CA.

[21] Kris Tiri and Ingrid Verbauwhede, “A Logic Level Design Methodology for a Secure DPA Resistant ASIC or FPGA Implementation”, in *DATE’04*. February 2004, pp. 246–251, IEEE Computer Society, Paris, France.

[22] Sylvain Guilley, Sumanta Chaudhuri, Laurent Sauvage, Tarik Graba, Jean-Luc Danger, Philippe Hoogvorst, Ving-Nga Vong, and Maxime Nassar, “Shall we trust WDDL?”, in *Future of Trust in Computing*, Berlin, Germany, jun 2008, vol. 2.