INTERNATIONAL JOURNAL OF RESEARCH
In Advanced Engineering Technologies

Volume 6, Issue 1 JUN 2017

REAL-TIME IMPLEMENTATION OF SIDE-CHANNEL PASSIVE ATTACKS ON CRYPTOGRAPHIC HARDWARE USING FPGA

S.PRADEEP[1] **J.LINGAIAH**[2]

pradeeep505@gmail.com[1]

[1]PG Scholar, Dept of ECE, Arjun College of Technology And Sciences, Mount Opera Premises BataSingaram, Hayath Nagar, RangaReddy, Telangana, India

[2]Assistant Professor, Guide, Dept of ECE, Arjun College of Technology And Sciences, Mount Opera Premises BataSingaram, Hayath Nagar, RangaReddy, Telangana, India

Abstract

Cryptanalysis is the field of research where the cryptographic algorithms are studied to break them and further extract the information. This subject gains more and more power with high speed running processors and FPGAs as the time required to iterate for millions of key word (password) combinations comes down with high speed running hardware. There are several types of cryptanalysis algorithms apart from brute force attacks on dictionary based searching. In recent days the new class of cryptanalysis algorithms is evolved based on clock violations and metastable conditions of flip-flops which are part of encryption algorithms.

In this project an FPGA based test bed is realized for injecting faults through clock glitches, to result in setup and hold violations. The UART interface is realized on FPGA to provide PC based controlling for this fault injection. The pre-build serial International Data Encryption (IDEA) algorithm synthesis models (.ngc files) will be used as test encryption algorithm. The Xilinx Digital clock manager (DCM) component will be used for generation clocks of different frequencies and phase shifts. The encryption module output with faults introduced and without fault introduced is compared as a function of ratio of used clock frequency and maximum frequency of operation reported by synthesis tool. The modules for clock generation, clock switching, interface adopter to IDEA core and UART interface will be realized and tested in FPGA hardware in integrated form.

From PC using HyperTerminal commands will be sent to FPGA firmware. Xilinx simulation and synthesis tools are used to this project. Xilinx Spartan family FPGA board along with serial communication with PC will be used for hardware level testing. Xilinx chip scope tools will be used for verifying the output at various levels in FPGA hardware.

1. INTRODUCTION

Cryptanalysis is the field of research where the cryptographic algorithms are studied to break them and further extract the information. This subject gains more and more power with high speed running processors and FPGAs as the time required to iterate for millions of key word (password) combinations comes down with high speed running hardware. There are several types of cryptanalysis algorithms apart from brute force attacks on dictionary based searching. In recent days the new class of cryptanalysis algorithms is evolved based on clock violations and meta stable conditions of flip-flops which are part of

encryption algorithms. Current problem is that the Attack technologies are being constantly improved and there is growing demand for secure chips. One of the proposed solutions is to use abnormal working conditions to generate malfunctions in the system which provides a way to analyze the susceptibility of FPGA to faults.

The main aim of this paper is to inject the faults into FPGA based logics using clock glitching mechanism and to thoroughly analyze the impact of injected faults on the operation of the circuit logic. A long term objective of our research is to develop an efficient method for protecting FPGA-based implementations of cryptographic algorithms through effective concurrent testing of various types of faults, including faults injected by the attackers. An essential part of this research is to develop a method and tool for the evaluation of susceptibility of FPGA based circuits to fault injection attacks. In this paper, we present such a method and tool. It allows us to examine an FPGA-based circuit, in particular an implementation of a cryptographic algorithm, subjected to a fault injection attack based on clock glitching. The effectiveness of the proposed approach is assessed for the IDEA implementation.

One of the cryptographic algorithms most commonly used and implemented in hardware is the Advanced Encryption Standard (AES), approved as a standard for symmetric cryptography by NIST [1]. therefore, become a target of various attacks, including those based on fault injection. Saha et al. report an attack through fault injection based on clock glitching. Such a technique allows the attacker to determine the encryption key from the real life FPGA-based AES implementation in about 400 seconds. Several concurrent checking methods and techniques for protecting hardware implementations of the AES algorithm against fault injection attacks have been proposed.

The effectiveness of these methods and techniques has been evaluated in different ways –in most cases through theoretical analyses and simulations. Testing against real life fault injection attacks has been uncommon. A long term objective of our research is to develop an efficient method for protecting FPGA-based implementations of cryptographic algorithms through effective concurrent testing of various types of faults, including faults injected by the attackers. An essential part of this research is to develop a method and tool for the evaluation of susceptibility of FPGA based circuits to fault injection attacks. In this paper, we present such a method and tool. It allows us to examine an FPGA-based circuit, in particular an implementation of a cryptographic algorithm, subjected to a fault injection attack based on clock glitching. The effectiveness of the proposed approach is assessed for the AES implementation

2. LITERATURE SURVEY

As mentioned above, the research on fault attacks started around two decades ago with the seminal paper of Boneh, DeMillo and Lipton. They showed the potential of this type of attacks assuming that they are possible. However, it took several years for a publication on an actual practical fault attack by Aumuller et al.. The authors published in 2002 one of the first practical works on fault analysis, in which they describe a real-life scenario of the impact of injecting glitches in the clock and voltage lines of a cryptographic chip.

Van Wouden berg et al. describes a real attack scenario for an Optical Fault Injection attack where a laser is used to induce faults. The authors also introduce the practical problem of

setting the parameters for fault injection and they briefly discuss the lack of methodology.

Recently, a paper of Boix Carpi et al. proposed a new algorithm that is a tailored search strategy but it is especially developed for finding the best choice of parameters for glitching. The authors also experiment with genetic algorithm and suggest investigating it further.

3. EXPERIMENTAL SETUP

The circuit under test (CUT) and the tester are both implemented on a low cost FPGA Spartan 3E development board. I used VHDL for defining custom components and Xilinx chip scope which is the system on-chip building tool, for creating standard library components and connections.

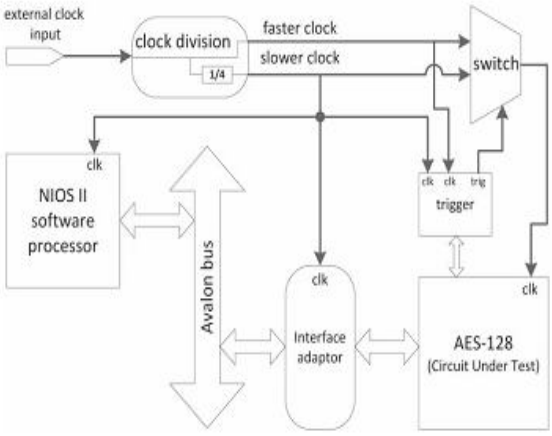

Figure 1: Experimental setup

A simplified diagram of experimental setup implemented in the device is shown in above Fig 1.

3.1Analysis of IDEA

In IDEA algorithm, I taken input text of size 64 bits at a time and divide it in evenly; i.e., 64 bit plain text is divided into 4 sub-blocks, each of 16 bits in size. The basic operations needed in the entire process for 8 rounds are 1. Multiplication modulo 2^16 +1.

2. Addition modulo 2^16.

3. Bitwise XOR.

And, operations needed in the OUTPUT TRANSFORMATION phase – 1. Multiplication module 2^16 +1. 2. Addition modulo 2^16. All the above mentioned operations are performed on 16 bit sub-blocks. For simplicity of expressing the operations. Now, let us take a look on the key generation for the encryption process while using the 25-bit circular left shift operation on the original key, it produce other subsequent sub-keys, used in different rounds [2]. For instance, among the total no. of 52 keys- Sub-key Z1 is having first 16bits of the original key, sub-key Z2 is having the next 16 bits, and so on till sub-key Z6; i.e., for ROUND1, sub-keys Z1 to Z6 use first 16x6= 96 bits of the original cipher key. In the ROUND2, sub-key Z7 &Z8 take the rest of the bits (bits 97 to 128) of the original cipher key. Then we perform circular left shift (by 25bits) operation on the original key.

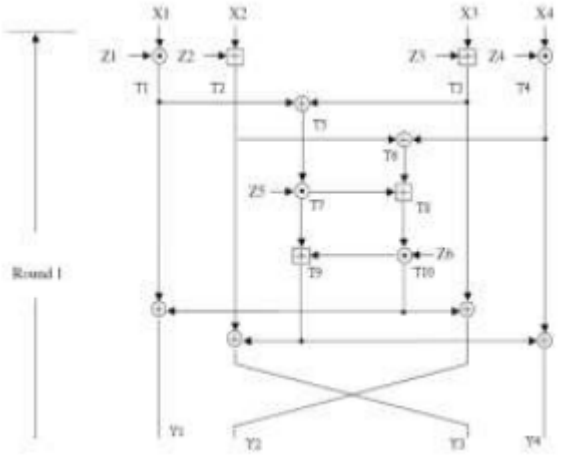

Figure 2: IDEA Encryption/Decryption sub key generation Architecture

As a result the 26th bit of the original key shifted to the first position and becomes the first bit (of the new shifted key) and the 25th bit of the original key is moves to the last position and becomes the 128th bit (after first shift). This process continues till ROUND8, and also in the OUTPUTPUT TRANSFORMATION phase; i.e., after the ROUND8, the key is again shifted left by 25 bits and the first 64 bits of the shifted key is taken for use, and used as sub-keys Z49 to Z52 in the OUTPUT TRANSFORMATIONMATION phase Output transformation stage The final round of IDEA algorithm is also called output transformation stage. It only uses 4 sub-keys. The block diagram of final round is given below. The VHDL code for IDEA final round module is given.

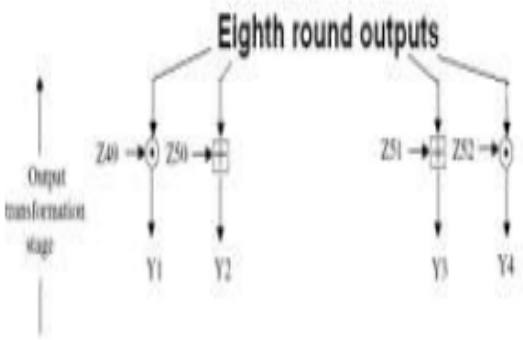

Figure 3: Block diagram of output transformation stage

The general IDEA architecture uses eight rounds with total 48 keys and final output transformation round with 4 sub-keys is implemented in VHDL using structural modeling style. The IDEA Decryption module also uses the same hardware, but the Decryption sub-keys are different. The encryption followed by decryption module is used for testing the complete IDEA algorithm with the following input. Main Key: Z = (5a14 fb3e 021c 79e0 6081 46a0 117b ff03) 64-bit plaintext: X = (X1, X2, X3, X4) = (7fa9, 1c3, ffb3, df05) The same text is used in simulating the other IDEA architectures.

3.2 Fault injection

The basic idea of our implementation of the fault injection based on clock glitching is to switch from a normal operation clock to a faster clock; so that one clock cycle is slightly shorter than CUT can handle. This idea is depicted in Figure 4. In order to generate single faults, the frequency of the faster clock has to be adjusted very precisely, more accurately than can be achieved using an on-chip PLL circuitry for clock generation or phase shifting.

An external clock signal generated by Tektronix AWG 5002B Arbitrary Waveform Generator is used instead. The external clock is going to feeds an internal PLL circuitry where it is divided by 4 to produce the slower clock. It also passes through unchanged to produce the faster (high speed) clock. To switch clocks, I using a Clock Control Block, the dedicated clock management built-in component available in the device. The result of the operations on clock signals is shown in Figure 4, as “output clock”. The last trace in Figure 4 is the real output clock registered by the 1 GHz Tektronix oscilloscope. The faster clock frequency is 150 MHz and it can be noticed that the signal is not distorted too much. Moreover, additional measurements, made by the MXG-9810AVolcraft frequency counter show that the faster clock has the same frequency (with accuracy of 1 Hz) as the clock supplied to the FPGA by the waveform generator.

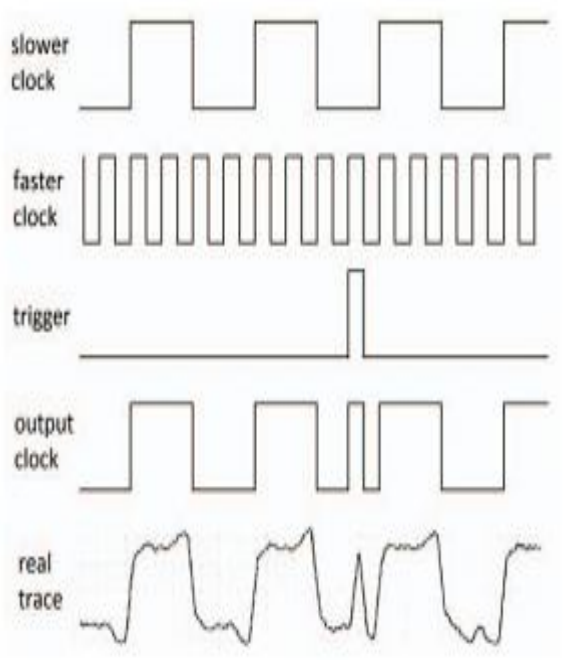

Figure 4. Clock glitch generation

3.3 Circuit under Test

To keep the circuit under test simple and at the same time relevant to our study, we have selected an iterative implementation of the Advanced Encryption Standard (AES) with a 128 bit key and without the decryption data path. We have chosen an iterative implementation, not a pipelined one, because it can be operated in more secure modes, like Cipher Block Chaining Mode or Output Feedback Mode. The encryption operation of AES consists of 9 identical rounds and a slightly modified 10th round, with one transformation omitted. In our implementation, the round keys are generated and stored in internal FPGA memory before the encryption procedure begins.

As reported, the encryption key can be deduced if a small number of faults are injected just before the 8th round of AES operation. To implement this kind of fault injection, the trigger unit is fed by the round counter included in the AES module. After detecting the beginning of the 6th round, the trigger unit counts two cycles of the faster clock and switches the Clock Control Block, so that to produce the faster clock pulse for one cycle in the 7th round. The maximum operation frequency for our AES module, reported by the Time Quest timing analyzer, varies from 166.6 MHz for 0°C to 151.0 MHz for 85°C.

4. RESULTS

The floating point adder Verilog HDL Modules have successfully simulated and verified using Modelsim6.4b and synthesized using Xilinxise10.1.

SIMULATION RESULT:

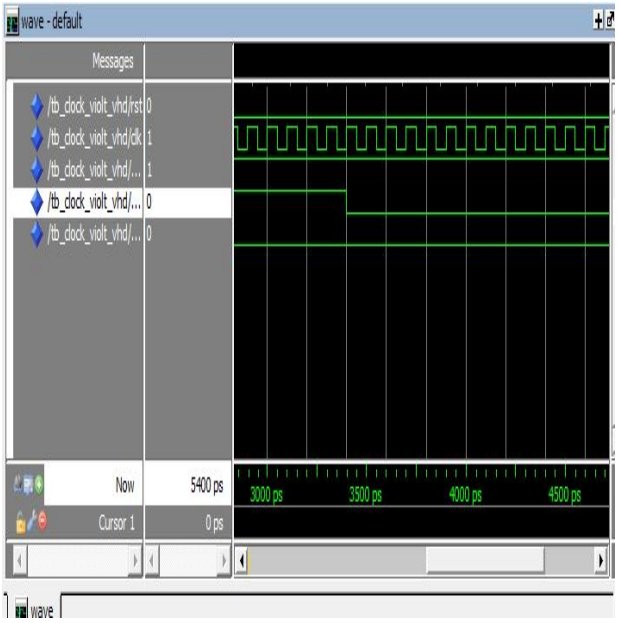

Figure5: Simulation for clock-violt

SYNTHESIS RESULTS:

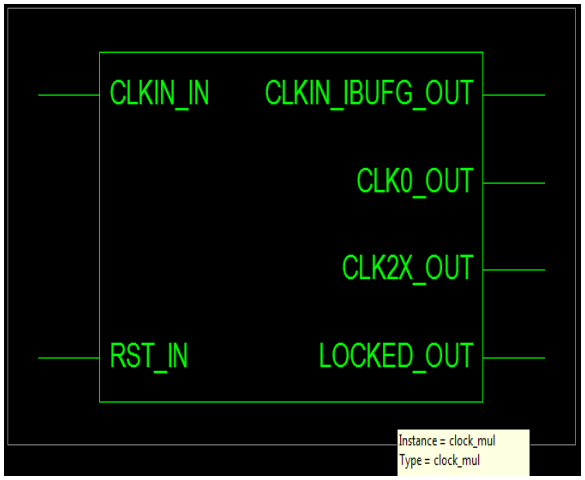

Figure 6: RTL for clock-mul

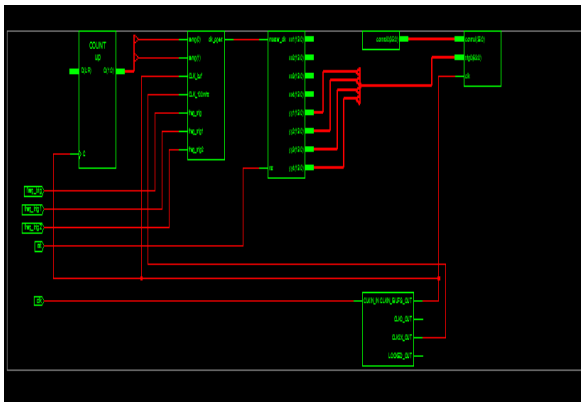

Figure 7: Schematic for clock-violt

DESIGN SUMMARY:

Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slices	536	3584	14%
Number of Slice Flip Flops	204	7168	2%
Number of 4 input LUTs	1025	7168	14%
Number of bonded IOBs	5	221	2%
Number of MULT18X18s	6	16	37%
Number of GCLKs	3	8	37%
Number of DCMs	1	4	25%

Figure 8: Device utilization for clock-violt

5. CONCLUSION

The presented method and tool for injecting faults in an FPGA-based circuit, based on clock glitching, has some unique features that allow us to thoroughly examine and analyze the impact of such faults on the operation of the circuit. In particular, through precise adjustment of the frequency of an external clock generator, we can control the number of faults occurring at the output of the circuit under test. The results obtained by applying the proposed technique to the AES implementation lead to a number of practical guidelines that may be essential when planning experimental studies on fault injection in FPGA-based circuits. They indicate that the impact of temperature on the outcome of the fault injection experiment based on clock glitching is not negligible –it affects the number of faults produced at the output of the circuit.

REFERENCES

[1] “Federal Information Processing Standards Publication 197— Announcing the Advances Encryption Standard (AES),” US National Institute of Standards and Technology, 2001.

[2] D. Saha, D. Mukhopadhyay and D. RoyChowdhury, "A Diagonal Fault Attack on the Advanced Encryption Standard", IACR Cryptology ePrint Archive, vol. 2009 , p. 581, 2009.

[3] K. Wu, R. Karri, G. Kuznetsov and M. Goessel, "Low Cost Concurrent Error Detection for the Advanced Encryption Standard", Proc. International Test Conference, pp 1242-1248, 2004.

[4] C.H. Yen and B.F. Wu, "Simple Error Detection Methods for Hardware Implementation of Advanced Encryption

Standard", IEEE Trans. on Computers, vol. 55, no. 6, pp. 720-731, 2006.
[5] K. Bousselam, G. Di Natale, M-L. Flottes, B. Rouzeyre, "Evaluation of concurrent error detection techniques on the Advanced Encryption Standard", Proc. 16th IEEE On-Line Testing Symposium, 2010.
[6] J. Balasch, B. Gierlichs and I. Verbauwhede, "An In-depth and Blackbox Characterization of the Effects of Clock Glitches on 8-bit MCUs", Proc. Workshop on Fault Diagnosis and Tolerance in Cryptography, 2011.
[7] Thomas Ruschival, "AES 128/192/256 (ECB) AVALON®-MM SLAVE", 2009, www.ruschival.de/wp-content/uploads/2009/05/avs_aes.pdf