CRYPTOGRAPHIC DESIGN AGAINST HTH VIRUS USING COUNTERFEIT LOGIC

***Ahamada Shreen J, ** Prem Ananth R**
** M.E VLSI DESIGN, SCAD College of Engineering & Technology, Cheranmahadevi, Tamilnadu,*
*** Assistant Professor, SCAD College of Engineering & Technology, Cheranmahadevi, Tamilnadu*

ABSTRACT

While research in security and its implementation in hardware has been on-going for many decades, the concept of hardware security was not formally introduced until the emergence of hardware Trojans and their detection methods. For most of existing Trojan detection methods, scalability becomes a concern. Therefore, researchers started to look into post-deployment methods leveraging post-deployment side-channel fingerprinting and on-chip equivalence checking. The key idea here was that stealthy hardware Trojans may easily evade detection methods during testing stage but, if triggered, they will cause large impact to side-channel fingerprinting or to circuit functionality. To increase HT detection sensitivity, this paper introduces the low-level HTH protection scheme by filling the unused resources of the FPGA with low-level counterfeit logic (LLCL).The proposed scheme identifies unused resources within the FPGA device and propose counterfeit logic cells for different resources of FPGAs. The proposed scheme significantly reduces the chance of application space HTH attacks by giving no free configurable resource for HTHs insertion in the design's bit stream. The malicious inclusion can either disturb the functionality of the original design, incur device fluctuations such as warming up or transistor aging.The proposed system can detect HTs, even if the HT size is small. Simulation results show that the proposed method achieves up to 100% HT detection rate.

Keywords: *limestone, cement production, SWIR, band ratio, Landsat 8, network analysis*

INTRODUCTION

A Trojan Horse is full of as much trickery as the mythological Trojan Horse it was named after. Those on the receiving end of a Trojan Horse are usually tricked into opening them because they appear to be receiving legitimate software or files from a legitimate source. When a Trojan is activated on your computer, the results can vary. Some Trojans are designed to be more annoying than malicious or they can cause serious damage by deleting files and destroying information on your system. Trojans are also known to create a backdoor on your computer that gives malicious users access to your system, possibly allowing confidential or personal information to be compromised. Unlike viruses and worms, Trojans do not reproduce by infecting other files nor do they self-replicate.

The main objective of this type of malware is to install other applications on the infected computer, so it can be controlled from other computers. Trojans do not spread by themselves, and as their name suggests, like the astute Greeks in their attack on Troy, these malicious codes reach computers in the guise of an apparently harmless program, which, in many cases, when executed releases a second program, the Trojan itself. Currently, the percentage of malware traffic represented by the Trojans worldwide is: Worm: 14.04%.

EXISTING METHODOLOGIES

Effects may range from a subtle degradation of service through to a complete and permanent shut-down of a system. Hardware Trojans can affect a system by themselves alone, or provide a foothold for software based attacks [4], where colluding software is aware of the inserted Trojan. Hardware Trojans most often remain dormant and are only activated when triggered by, for example, the passing of time, some specific activity, or external events.

A Hardware Trojan might be as simple as a paragraph change in the specification, an extra source-code line in the Hardware Description Language (HDL), a modification of the silicon die at the fabrication plant, or just a modulation of the CMOS geometries used. Ensuring an IC is authentic and does not contain any Trojans is a very difficult problem [5].

Hardware Trojans are poorly observable during traditional IC design-verification and testing phases; their payload and trigger states are difficult to find, becoming exponential in the number of circuit nodes [3]. Most ASIC designs are too large for either exhaustive testing or formal verification, so it is likely that many Trojans will never be detected. The life-cycle stage at which a Trojan is inserted and its logical structure also affects the effectiveness of existing detection methods.

Once detected, a Hardware Trojan may be removed from a design (if detected in the RTL), the IC could be set aside so that it is not used, or the IC may still be used, operating in the presence of the Hardware Trojan . Depending on the detection mechanism used, a Hardware Trojan may be either definitively identified, or a statistical measure may be provided indicating the probability that the design or IC has been tampered with. Traditional IC test and verification is targeted at performing acceptance tests and ensuring an IC performs as specified. Generally, however, it does not test for the addition of extra functionality. There is still a very real possibility that a Hardware Trojan will not be picked up during testing, but will be activated once the chip is in use. Abramovici & Bradley [2] provide a strong argument that “we cannot guarantee that ICs deployed in the field are Trojan-free”.

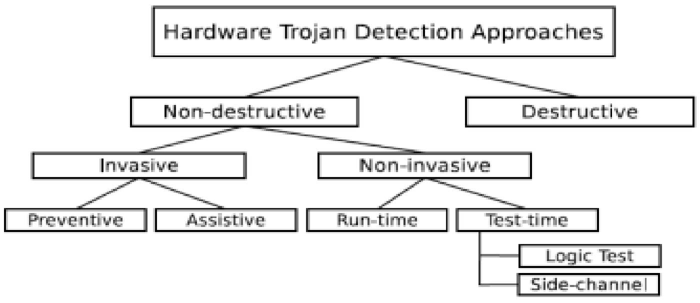

Fig. 1 Hardware Trojan Detection Approaches

PROPOSED SYSTEM

A low-level HTH protection scheme by filling the unused resources of the FPGA with low-level counterfeit logic (LLCL). The proposed scheme identifies unused resources within the FPGA device and proposes counterfeit logic cells for different resources of FPGAs. The proposed scheme significantly reduces the chance of application space HTH attacks by giving no free configurable resource for HTHs insertion in the design's bit stream. The malicious inclusion can either disturb the functionality of the original design, incur device fluctuations such as warming up or transistor aging. The proposed scheme does not impose any power or performance overheads as compared to the original non protected designs and also achieves high resource utilization. The proposed system can detect HTs, even if the HT size is small. Simulation results show that the proposed method achieves up to 98% HT detection rate

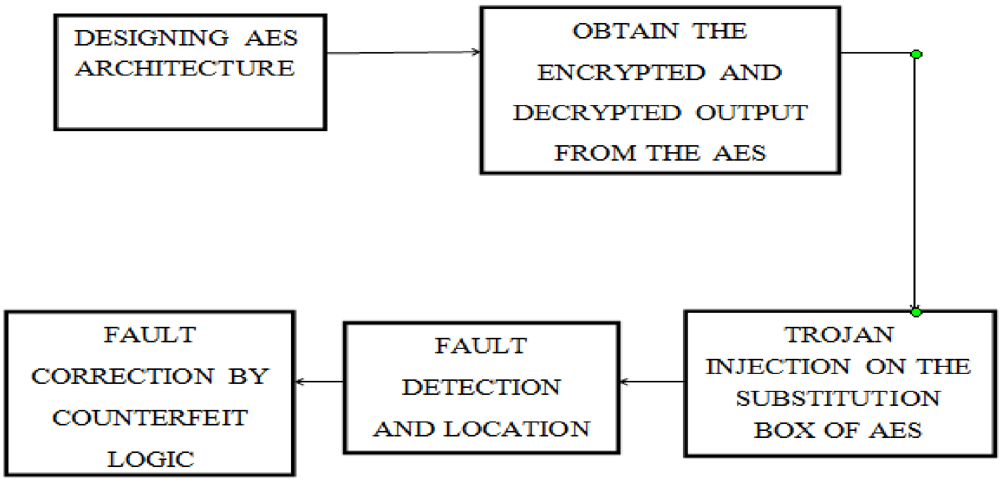
DESIGNING AES ARCHITECTURE
OBTAIN THE ENCRYPTED AND DECRYPTED OUTPUT FROM THE AES
TROJAN INJECTION ON THE SUBSTITUTION BOX OF AES
FAULT DETECTION AND LOCATION
FAULT CORRECTION BY COUNTERFEIT LOGIC

Fig. 2 Modules of the Proposed System

IMPLEMENTATION WORK

A. Modular S-Box

This section illustrates the steps involved in constructing the multiplicative inverse module for the S-Box using composite field arithmetic. Since both the SubByte and

InvSubByte transformation are similar other than their operations which involve the Affine Transformation and its inverse, therefore only the implementation of the SubByte operation will be discussed in this paper. The multiplicative inverse computation will first be covered and the affine transformation will then follow to complete the methodology involved for constructing the S-Box for the SubByte operation. For the InvSubByte operation, the reader can reuse multiplicative inversion module and combine it with the Inverse Affine Transformation,.The individual bits in a byte representing a GF(28) element can be viewed as coefficients to each power term in the GF(28) polynomial. For instance, {10001011}2 is representing the polynomial q*7 + q3 + q + 1* in GF(28). From [2], it is stated that any arbitrary polynomial can be represented as *bx* + *c*, given an irreducible polynomial of *x2* + *Ax* + *B*. Thus, element in GF(28) may be represented as *bx* + *c* where *b* is the most significant nibble while *c* is the least significant nibble. From here, the multiplicative inverse can be computed using the equation below.

$$(bx+c)^{-1} = b\left(b^2B + bcA + c^2\right)^{-1}x + (c+bA)\left(b^2B + bcA + c^2\right)^{-1}$$

From [1], the irreducible polynomial that was selected was *x2* + *x* + *λ*. Since *A* = *1* and *B* = *λ*, then the equation could be simplified to the form as shown below

$$(bx+c)^{-1} = b\left(b^2\lambda + c(b+c)\right)^{-1}x + (c+b)\left(b^2\lambda + c(b+c)\right)^{-1}$$

The above equation indicates that there are multiply, addition, squaring and multiplication inversion in GF(24) operations in Galois Field. Each of these operators can be transformed into individual blocks when constructing the circuit for computing the multiplicative inverse. From this simplified equation, the multiplicative inverse circuit GF(28) .

.

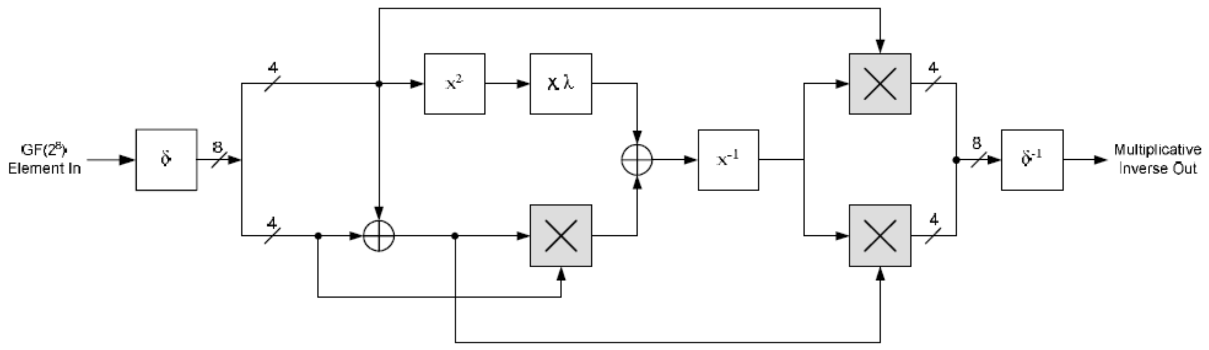

Fig. 3 Moduler S-Box

RESULTS AND DISCUSSION

A. ENCRYPTION IN AES ARCHITECTURE

Fig 4 shows the process of encryption in AES architecture. The process is carried out by 128 bit plain text such as “SHABEENA” and the key whose size is also 128 bit such as

"VLSI". The encryption process produce the cipher text of 128 bits after 10 rounds of processing, as shown above.,

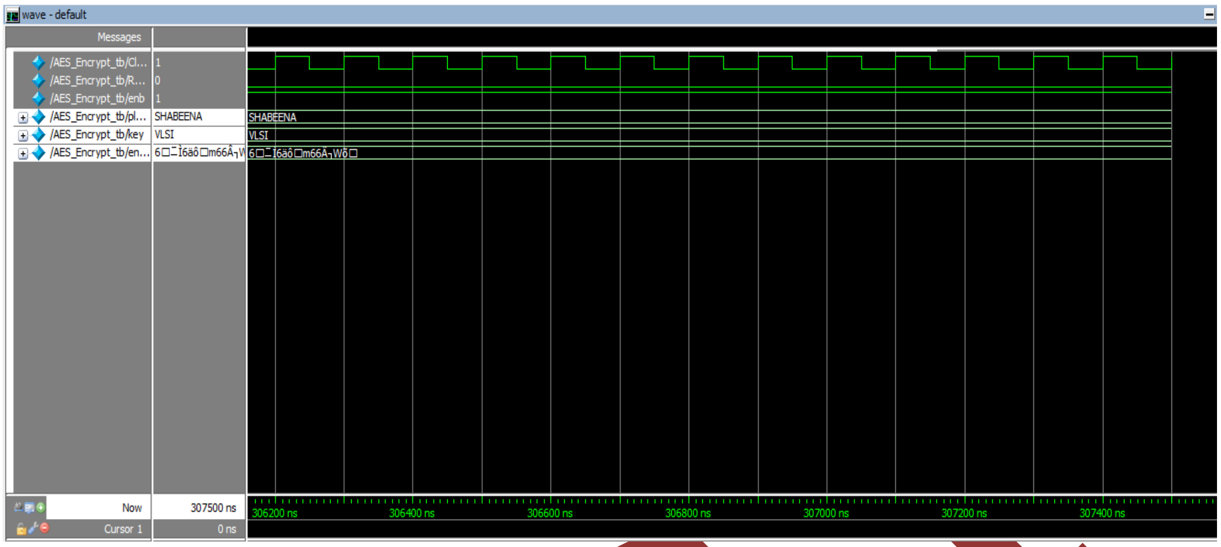

Fig 4 Encryption in AES Architecture

B. DECRYTION IN AES ARCHITECTURE

Fig 5 shows the process of decryption in AES architecture. The process is carried out by 128 bit cipher text and the key whose size is also 128 bit such as "VLSI", which results in the retrieval of plain text such as "SHABEENA". The decryption process in AES architecture is carried out as same as that of encryption process, but in the exact opposite direction

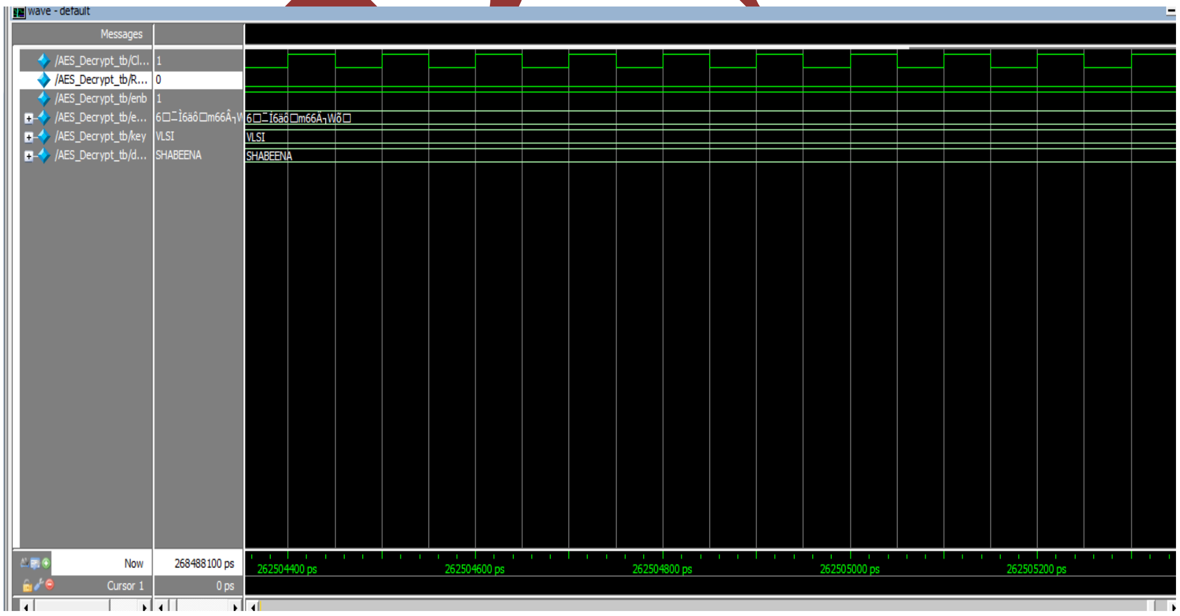

Fig 5 DECRYTION IN AES ARCHITECTURE

Table 1: POWER ANALYZER TABLE

Method	**AES**	**Modular S-Box**
Thermal Power Dissipation	**180.70**	**155.53**
Core Static Power Dissipation	**89.02**	**80.07**

CONCLUSION

In this project, a low-level HTH protection scheme by filling the unused resources of the FPGA with low-level counterfeit logic (LLCL) is proposed. The proposed scheme identifies unused resources within the FPGA device and proposes counterfeit logic cells for different resources of FPGAs. This scheme significantly reduces the chance of application space HTH attacks by giving no free configurable resource for HTHs insertion in the design's bit stream. The malicious inclusion can either disturb the functionality of the original design, incur device fluctuations such as warming up or transistor aging. In the S-box of AES, the hexadecimal values that are given to each input unit get scrambled due to the Trojan infection. The values assigned in the S-box are getting confused and the encrypted output gets modified due to the Trojan attack, and the counterfeit circuit helps to correct the faults that are caused by the Trojan in the S-box of AES. The proposed protection LLCL scheme does not alter the placement and routing of the original design and only exploits the unused configurable resources in the FPGA. The proposed system can detect HTs, even if the HT size is small. Simulation results show that the proposed method achieves up to 98% HT detection rate.

REFERENCES

[1] Agrawal et al.D, (2007), ‘‘Trojan Detection Using IC Fingerprinting,’’Proc. IEEE Symp. Security and Privacy (SP 07), IEEE CS Press, pp. 296-310.

[2]Ahmed Aliyu, Abdulaziz Bello, Usman Joda Mohammed Ibrahim Hussaini Alhassan,(2014),” Hardware Trojan Model For Attack And Detection Techniques”, INTERNATIONAL JOURNAL OF SCIENTIFIC & TECHNOLOGY RESEARCH VOLUME 3, ISSUE 3, ISSN 2277-8616,Pages: 102-105.

[3]Asadi.H and Tahoori.M.B,(2007), “Analytical techniques for soft error rate modeling and mitigation of FPGA-based designs,” *IEEE Trans. Very Large Scale Integr.*, vol. 15, no. 12, pp. 1320–1331.

[4]Behnam Khaleghi, Ali Ahari, Hossein Asadi, and Siavash Bayat-Sarmadi,(2015),” FPGA-Based Protection Scheme against Hardware Trojan Horse Insertion Using Dummy Logic”, IEEE EMBEDDED SYSTEMS LETTERS, VOL. 7, NO. 2, Pg 46-50.

[5]Canright.D and Osvik.D.A,(2009), "A More Compact AES," Selected Areas in Cryptography, pp. 157-169, Springer-Verlag.

[6]Chakraborty.R.S, Narasimhan.S, and Bhunia.S,(2009) "Hardware trojan: Threats and emerging solutions," in *Proc. IEEE Int. High Level Design Validation Test Workshop (HLDVT)*, pp. 166–171, IEEE.

[7]Chakraborty.R.S, Paul.S, and Bhunia.S,(2008), ''On-Demand Transparency for Improving Hardware Trojan Detectability,'' Proc. IEEE Int'l Workshop Hardware-Oriented Security and Trust (HOST 08), IEEE CS Press, pp. 48-50.