Investigating TERO for Hardware Trojan Horse Detection

Paris Kitsos[¶]
Computer and Informatics Engineering Department
Technological Educational Institute of Western Greece
Antirrion, Greece
e-mail: pkitsos@teimes.gr

Artemios G. Voyiatzis
Industrial Systems Institute, "Athena" Research and Innovation Center in ICT and Knowledge Technologies
Platani Patras, GR-26504, Greece
e-mail: bogart@isi.gr

***Abstract*—The Transient Effect Ring Oscillator (TERO) was initially proposed for TRNG designs and lately for implementing Physically Uncloneable Function (PUF) designs. In this work, the effectiveness of TERO for hardware Trojan horse detection is examined. For this purpose, the implementation of the SNOW3G stream cipher and three example Trojans are used. Experiments and comparisons are reported in terms of the frequency count of the Trojan-free and the Trojan-infected (modified) circuits containing TEROs of variable lengths. Our findings indicate that a TERO can be a sensitive sensor but still it cannot provide reliable detection in all cases.**

Keywords—hardware Trojan horse detection; SNOW3G stream cipher; Transient Effect Ring Oscillator (TERO); hardware malware; Spartan 6 (SAKURA G board) FPGA

I. INTRODUCTION

The Transient Effect Ring Oscillator (TERO) is a circuit that oscillates due to its inherent logic. The oscillation frequency depends on the exact components and the size of a circuit similarily to the case of a Ring Oscillator (RO). TERO was initially proposed for implementing a True Random Number Generator [1]. Recently, it was proposed for implementing also a Physically Uncloneable Function (PUF) [2]. In this work, we propose a novel use of TERO for hardware Trojan detection and we explore its applicability and efficiency as a extra tool for hardware Trojan horse detection.

II. TERO CIRCUIT

A Transient Effect Ring Oscillator (TERO) is composed of an SR flip-flop implemented with two XOR gates and two AND gates [1]. This architecture has two control signals, for start and reset. The correct place-and-routing for a TERO is important so as to ensure the same length of the interconnections between the XOR gates.

In this work, we have used a simpler TERO architecture, where the XOR and AND gates are merged into NAND gates with some inverters in the feedback loop, as depicted in Fig. 1. The advantage of this approach is that only one control signal is used either for resetting or oscillating the TERO circuit. The reset occurs when the control signal, *ctrl*, is set to '0' and drives the loop to the same initial conditions before generating its output. When the control signal changes from '0' to '1', the TERO circuit starts to oscillate. An asynchronous counter (Counter) was used so as to measure the TERO frequency.

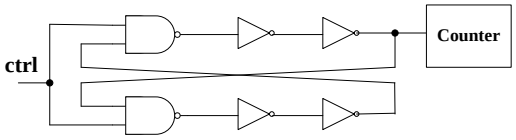

Fig. 1. TERO circuit

III. EXPERIMENTAL SETUP

In order to investigate the effectiveness of the TERO, we realized three hardware Trojan horses. The first one (Trojan1), a combinational circuit, is formed as a tree of AND gates and the output of the tree is fed into a XOR gate that drives the system reset signal. Trojan1 reads bits 24-31 of the output. If the bits are equal to 0xFF, the Trojan is activated and deactivates the reset signal.

Trojan2 is a time bomb. It consists of a simple counter and a AND tree that reads bits 13-16 of the cipher's output and the tree output drives the enable signal of the Trojan counter. If the AND tree counts 100 sequences of "111" at the cipher's output, then Trojan2 deactivates the SNOW3G reset signal.

Trojan3 consists of two AND trees and two asynchronous counters. The first AND tree focuses on bits 13-16 of the cipher output and it is used so as to activate the first counter (*counter1*). If a sequence of "1111" occurs, then *counter1* is activated. Every second activation of *counter1*, Trojan3 outputs and activation signal (*tmp_load*) that is used so as to trigger the second counter (*counter2*). The *tmp_load* signal is combined with three internal bits and through the second AND tree is used for activating the second counter. Then, after 62 pulses, *counter2* deactivates the cipher reset signal.

Following the taxonomy of [3], we assume that the malicious circuit is inserted during the design phase, at the register transfer level and located at I/Os, with the aim to perform a denial-of-service (DoS) attack.

The three Trojans target the SNOW3G stream cipher [4]. The SNOW3G is a word-oriented stream cipher that generates a sequence of 32-bit words under the control of a 128-bit key and a 128-bit initialization variable. At first, a key initialization process is performed and the cipher is clocked without producing output. Then, the cipher operates in the key-generation mode and it produces a 32-bit ciphertext/plaintext word output in every clock cycle. The architecture of the SNOW3G cipher is depicted in Fig. 2. A cryptographic primitive is an attractive target for a Trojan due to the critical

¶ Collaborating Faculty with the Industrial Systems Institute, "Athena" RIC in ICT and Knowledge Technologies.

This work was partially supported by the GSRT Action "KRIPIS" with national and EU funds in the context of the research project "ISRTDI" and by COST Action IC1204 "TRUDEVICE".

information processed and the enormous search space for detecting. The Trojan payload may leak the key information, skip the cryptographic operations altogether, or perform a denial-of-service attack.

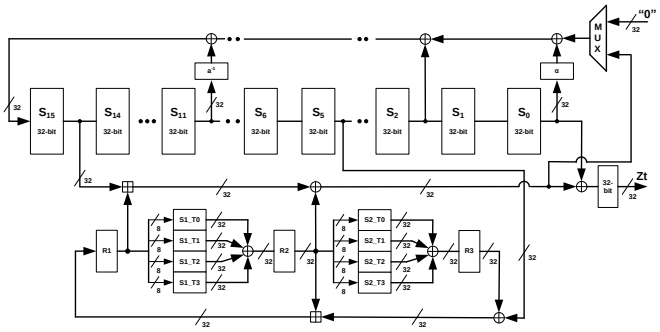

Fig. 2. SNOW3G hardware architecture[5]

A Spartan 6 (XC6SLX75-2CSG484C) FPGA, the base of the SAKURA G board, was used in our experiments. The setup consists of the design of SNOW3G cipher with TERO and the circuits of the hardware Trojan horses. We used the implementation file extracted by PleanAhead tool and especially the VHDL code of the Post-Place and Route simulation model. The model was simulated so as to derive the TERO oscillation frequency.

In our experiments, we used TEROs of different lengths so as to devise the optimal length for detecting the Trojan. The hardware Trojan horses occupied a small percentage of the available area and the TERO was placed in the circuit in a controllable fashion. In order to insert the hardware Trojan horses in a design implemented on FPGA, we use the hardware description language (HDL). While this method can be used to create many types of hardware Trojan horses, it is impossible to guarantee the exact place for the hardware Trojan horse insertion. If two systems are synthesized on the same FPGA board and they differ only to one hardware resource, the synthesis procedure will probably devise a completely different placing and routing.

In order to achieve efficient, fair, and, most importantly, accurate measurements, one must build designs with the same place and route for clean and infected SNOW3G. This can be accomplished using BEL and LOC placement constraints. For the case of TERO, we have used parameterized area constraints. There are four designs: a) a Trojan-free, containing SNOW3G and TERO), b) the Trojan-free plus Trojan1, c) the Trojan-free plus Trojan2, and d) the Trojan-free plus Trojan3.

IV. Results and Future Work

Snapshots of the four layouts are depicted in Fig. 3. The TERO layout is shown as a white trace. It can be seen that the same layout for the circuits SNOW3G and TERO were created. This means that the hardware resources of the identical circuits are placed and routed on the same locations on the FPGA. We decided to diffuse the Trojans around the cipher circuit and implement TERO in between the SNOW3G cipher and Trojans so as to better "sense" the process variations. In the sense that the greater distance between the counts means better reliability and detection sensitivity of designs with hardware Trojan horses, the best metric is the absolute difference of oscillation counts between the Trojan-free and the infected circuits.

Table I summarizes the performance of TERO. It is clear that TERO is more sensitive when small lengths are used. Its sensitivity decreases as the lengths increase and in the case of Trojan1, it cannot reliably detect the Trojan presence.

As a future work, we plan to implement SNOW3G as an IP core and experiment with TERO in SAKURA G and VC707 (Virtex 7) boards.

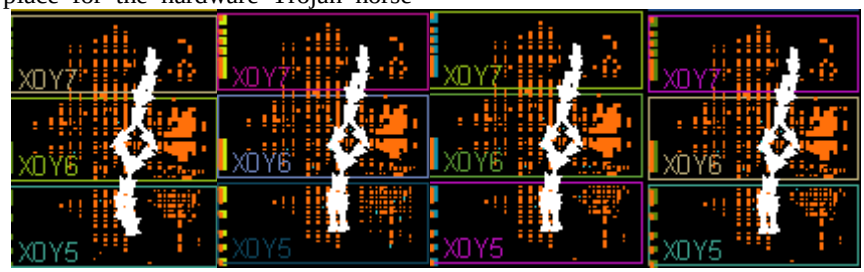

Fig. 3. Implementation layout of (a) SNOW3G andTERO (no Trojan); (b) infected with Trojan1 (c) infected with Trojan2; and (d) infected with Trojan3

TABLE I. Absolute Differences for TERO

TERO length	Abs. Diff (Tr. free-Tr.1)	Abs. Diff. (Tr. free-Tr.2)	Abs. Diff. (Tr. free-Tr.3
TERO-04	99	99	165
TERO-08	20	99	66
TERO-12	9	67	51
TERO-16	2	40	41
TERO-20	0	66	66
TERO-24	4	48	48

References

[1] M. Varchola and M. Drutarovsky, "New high entropy element for FPGA based true random number generator", in Proc. Int. Conf. CHES, Santa Barbara, 2010, pp. 351-365.

[2] L. Bossuet, X, Thuy Ngo, Z. Cherif, and V. Fischer, "A PUF based on a Transient Effect Ring Oscillator and insensitive to locking phenomenon", IEEE Trans. on Emerging Topics in Computing, Vol. 2, No. 1, pp. 30-36, March 2014.

[3] R. Karri, J. Rajendran, K. Rosenfeld and M. Tehranipoor, "Trustworthy hardware: Identifying and classifying hardware Trojans", IEEE Computer, October 2010, vol. 43, no. 10

[4] Specification of the 3GPP Confidentiality and Integrity Algorithms UEA2 & UIA2. Document 2: SNOW 3G Specification, ETSI/SAGE Specification, Version: 1.1 Date: 6th September 2006.

[5] P. Kitsos, G. Selimis, O. Koufopavlou, "High performance ASIC implementation of the SNOW 3G stream cipher", IFIP/IEEE VLSI-SOC 2008 - International Conference on Very Large Scale Integration (VLSI SOC), Rhodes Island, Greece, October 13-15, 2008.