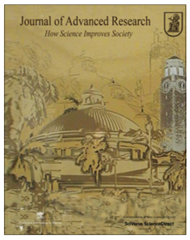
Cairo University

Journal of Advanced Research



ORIGINAL ARTICLE

System-level protection and hardware Trojan detection using weighted voting

Hany A.M. Amin [a,*], Yousra Alkabani [b], Gamal M.I. Selim [a]

[a] *Computer Engineering, Engineering & Technology Collage, AASTMT, Cairo, Egypt*
[b] *Computer & Systems Engineering, Faculty of Engineering, Ain Shams University, Cairo, Egypt*

ARTICLE INFO

Article history:
Received 16 September 2013
Received in revised form 27 November 2013
Accepted 28 November 2013
Available online 6 December 2013

Keywords:
Trojan
Third party IP
Attacker
Chips

ABSTRACT

The problem of hardware Trojans is becoming more serious especially with the widespread of fabless design houses and design reuse. Hardware Trojans can be embedded on chip during manufacturing or in third party intellectual property cores (IPs) during the design process. Recent research is performed to detect Trojans embedded at manufacturing time by comparing the suspected chip with a golden chip that is fully trusted. However, Trojan detection in third party IP cores is more challenging than other logic modules especially that there is no golden chip. This paper proposes a new methodology to detect/prevent hardware Trojans in third party IP cores. The method works by gradually building trust in suspected IP cores by comparing the outputs of different untrusted implementations of the same IP core. Simulation results show that our method achieves higher probability of Trojan detection over a naive implementation of simple voting on the output of different IP cores. In addition, experimental results show that the proposed method requires less hardware overhead when compared with a simple voting technique achieving the same degree of security.

Introduction

In the past two decades, security researches have focused on both network and information security and how to prevent cyber attacks. However, hardware Trojan Horses cause a deeper breach bypasses upper security layers and threaten all the entire critical infrastructures such as military infrastructure, financial systems and transportation vehicles. Hardware chips are becoming more vulnerable to malicious activities and alterations during both design and manufacturing phases.

In general, hardware Trojans try to bypass or destroy the three major security concerns (CIA) of any system by: leaking confidential information and secret keys covertly to the adversary (Confidentiality attack); changing the value of a certain register (Integrity attack); disabling, deranging or destroying the entire hardware or components of it (Availability attack).

Traditional Hardware testing strategies cannot effectively detect Trojans because the probability of triggering a hardware Trojan during functional testing is extremely low. Plus, the small Trojan size with respect to chip overall size reduces the Trojan impact on side channels such as static and dynamic power.

Hardware Trojans can be a simple modification to the original circuit as shown in Fig. 1; Adversary inserts a simple two input AND gate between the original circuit output and logical

* Corresponding author. Tel.: +20 1111007735.
E-mail address: hanyamin@gmail.com (H.A.M. Amin).
☆ Peer review under responsibility of Cairo University.

2090-1232 © 2013 Production and hosting by Elsevier B.V. on behalf of Cairo University.
http://dx.doi.org/10.1016/j.jare.2013.11.008

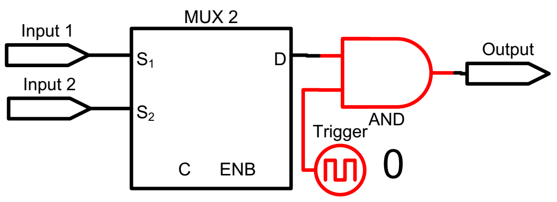


Fig. 1 "SAZ" hardware Trojan.

one. If Trojan is inactive, circuit will produce the real output; and if Trojan is triggered and becomes active, Input will be logical zero so circuit will produce "Zero" output disregarding original input value as explained in Eqs. (1) and (2). It is called SAZ Trojan (Stuck at Zero) as circuit output will stick at "Zero" if Trojan is activated.

$$X \cdot 1 = X \quad (1)$$

$$X \cdot 0 = 0 \quad (2)$$

Majority voting technique can be used for protection with no need for a fully trusted chip as shown in Fig. 2. We aim to produce a Trojan free output from infected IP cores. We use voting techniques for the output of odd number of multi-vendor IP cores trying to achieve negligible probability of infected output and report the infected IP core. Although the use of simple majority voting was suggested in other papers by Waksman and Sethumadhavan [1], it was not thoroughly evaluated using hardware implementation. In this paper, we evaluate the protection method based on the probability of Trojans detection, probability of false positives, and probability of false negatives. We also suggest an advanced voting technique based on giving a higher voting weight for trusted IP cores. We evaluate both the security properties and hardware overhead of both voting methods. Hardware overhead here means circuit area, circuit delay and Leaked power (see Fig. 3).

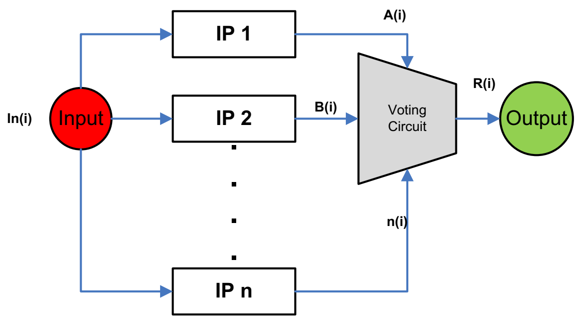


Fig. 2 Majority voting technique.

Hardware chips fabrication process contains two major steps: design (including IP, models, tools, and designers); and fabrication (including mask generation and packaging). In an ASIC design process, the IP core blocks and standard model cells which are used by the designer during the design process are considered untrusted, also hardware fabrication step may be considered untrusted because an attacker may replace Trojan logic for original ones or inject a Trojan into chip silicon mask.

The attacker is assumed to alter the design maliciously before or during fabrication, and detecting these alterations is extremely difficult, as detecting small malicious alteration is extremely harsh in today's high complex IP cores. Nano-meter physical inspection is very sophisticated and costs a lot. Trojans are activated under rare conditions so normal function testing is not sufficient to detect them.

It is mandatory to provide methods that resolve the trust issues among fabrication facilities, designers, and end users. Designers need to assure that their designs are not altered while maintaining fabrication facilities technology secrets and third party IP core design properties.

Related work

Many hardware Trojan detection methods have been developed to protect against Trojans [2–6]. These methods either try to detect the existence of a Trojan by analyzing side channels [7–13], or try to introduce architectural modifications to make Trojan insertion more difficult [14–17]. However, these methods mainly depend on comparing the suspected chip with a golden chip (a known trusted chip). In practice, a golden chip might not be available especially when using third party IP cores. Attempts to depend on using the system integrator's design specifications for comparisons were introduced by Zhang and Tehranipoor [18].

Logic duplication is proposed by Waksman and Sethumadhavan [1], where outputs from the modules are then checked cycle-by-cycle, they proposed to obfuscate and randomize the inputs to different hardware modules to misguide any Trojans and prevent it from recognizing triggers. They focused on proposing three main methods of hardware randomization that match with the three major types of Trojans triggers. Power reset obfuscates timing information to prevent units from detecting how long they have been powered on. Data obfuscation misguides infected units by using inputs encryption. Sequence breaking reorders micro-architectural events to handle Trojans triggered by control information. McIntyre et al. [19] utilize a method for dynamically evaluating the trust in hardware at run-time. They proposed using a multi-core processing system to take advantage of in-build redundancy, and the ability to discard cores if they are found to be untrusted. The variation in processes may be obtained from different compilation, implementation, or algorithms used. They have explored the effectiveness of dynamic distributed multicore trust determination by simultaneously executing a variant of the subtask on another core module to detect Trojans. The subtask scheduling is mandatory to coordinate the subtask variants produces both new learning with high confidence of core module trusts and high confidence of valid subtask execution results. Their scheduler is able to use learned core module trust to more efficiently execute needed jobs with increased throughput. Critical to their approach of dynamic trust determination are the generation and execution of functionally equivalent binary variants of a subtask. Baumgarten et al. [20] introduced using reconfigurable logic barriers within a design to prevent the activation and operation of hardware Trojans added during the manufacturing stage of an IC and then they evaluated the resiliency of their approach to Trojan detection. Their contributions include a combinational-locking scheme integrated into a standard CAD tool flow to prevent IC piracy, the first metering scheme that does not disclose the entire schematic to the foundry, and efficient node selection heuristics for maximizing security while minimizing associated overhead. Newgard and Hoffman [21] introduce a tightly coupled dual-processor lock-step configuration implemented inside an FPGA – an implementation of replication and voting at the macro-level. Both processors receive and process the same instructions at the same time. Hardware check logic examines and compares all bus control signals on every bus transaction. If an error is detected, the system is forced into an error recovery sequence. Beaumont et al. [22] run replica of a program on multiple processing elements to achieve protection form hardware Trojans. All the mentioned methods did not give too much attention to the voting technique among the duplicated logic gates. This paper mainly focuses on analyzing majority voting techniques among duplicated IP cores and presenting a new majority voting technique to achieve better hardware security performance.

Majority voting

Comparing the output, timing, and power consumption of a suspected chip with a trusted chip is the common way to detect hardware Trojans. However, this way cannot be used with third party IP cores as there is no golden IP core to compare with.

In this work, we eliminate the need to golden chip to detect a Trojan. Our main concern is dynamically protecting the chip from any suspicious activity. This is achieved by majority voting technique by using an odd number of untrusted IP cores from multiple vendors; the outputs from IP cores are validated on bit-by-bit basis by doing effective voting to produce the correct output. The main two benefits for using different implementations of IP cores are the following: (1) protecting against any functional disruptions using the duplicated logic and (2) protecting against (DoS) availability attacks by providing redundancy in operation of logic elements within the design.

Our countermeasure can be deployed at various levels from gate, RTL, logic design, functional modules, and IP cores, even though the IC and macro-level devices. The protection mechanisms rely on a non-collusion assumption among the duplicated IP cores within the design.

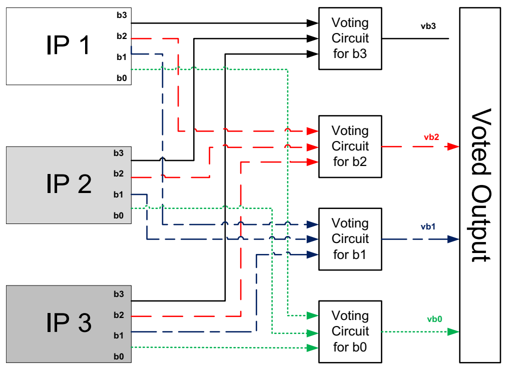


Fig. 3 Voting circuit.

Simple voting

Simple voting is a naive majority voting technique. By doing a bit level democratic majority voting among multiple IP cores outputs, we are producing the majority consensus result. Table 1 shows an example of simple voting on 3 IP cores, each

Table 1 Simple voting truth table.

IP1	IP2	IP3	SVR
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

IP core has 1 bit only. If the number of logical ones is greater than number of logical zeroes, the output will be logical one and vice versa. Table 1 describes the truth table of simple voting circuit (see Tables 2–4).

Simple democratic voting is producing efficient results in terms of security performance. It produces high Trojan detection percentage, low false positives and low false negatives. The main problem in this technique is the assurance of majority result. If some Trojans are fired on most of IP cores at the same time, the majority result will be infected. Fig. 4 explains the flowchart of simple voting technique. Eq. (3) explains a simple implementation of 1-bit simple voting logic circuit with using 3 different IP cores, below equations are conducted from truth table in Table 1, Vbx is the voted result of bit in position x while bx|1 is the bit in position x of the first IP core and WL is the word length for all three IP cores.

$$\begin{aligned}\mathrm{Vbx} = {} & (\mathrm{bx}|1' \cdot \mathrm{bx}|2 \cdot \mathrm{bx}|3) + (\mathrm{bx}|1 \cdot \mathrm{bx}|2' \cdot \mathrm{bx}|3) + (\mathrm{bx}|1 \\ & \cdot \mathrm{bx}|2 \cdot \mathrm{bx}|3') + (\mathrm{bx}|1 \cdot \mathrm{bx}|2 \cdot \mathrm{bx}|3), x \\ = {} & [0 \rightarrow \mathrm{WL}] \end{aligned} \tag{3}$$

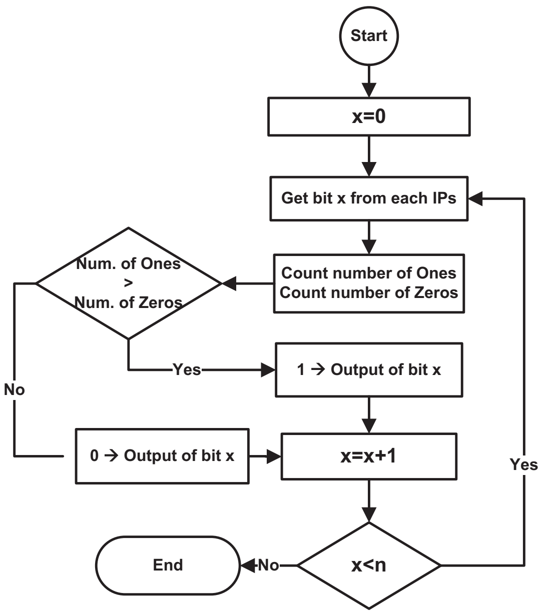


Fig. 4 Simple voting algorithm.

Weighted voting

The second proposed method is doing weighed majority voting among multiple IP cores. Voting algorithm selects the higher

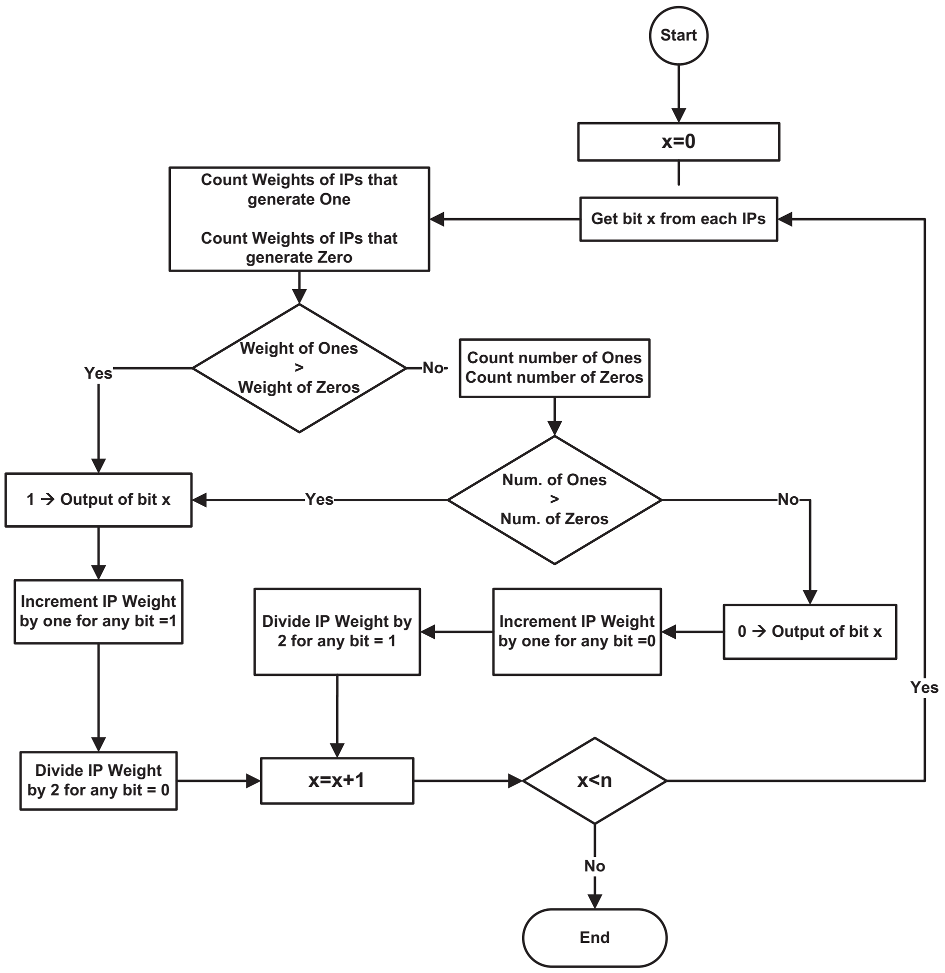

Start
x=0
Get bit x from each IPs
Count Weights of IPs that generate One
Count Weights of IPs that generate Zero
Weight of Ones > Weight of Zeros
Yes
No
Count number of Ones
Count number of Zeros
Num. of Ones > Num. of Zeros
Yes
No
1 → Output of bit x
0 → Output of bit x
Increment IP Weight by one for any bit =1
Divide IP Weight by 2 for any bit = 1
Increment IP Weight by one for any bit =0
Divide IP Weight by 2 for any bit = 0
x=x+1
x<n
Yes
No
End

Fig. 5 Weighted voting algorithm.

weighted IP core bit. Each IP core initial weight counter is zero as explained in Eq. (4). After each cycle, voting algorithm calculates sum of weights for each IP cores resulting logical one and sum of weights for each IP cores resulting logical zero. Then it compares both sums as explained in Eqs. (5) and (6). The higher sum will be considered the correct result bit and the voted output, and the lower sum is the infected IP cores bit. Weight counters are increased by one for all IP cores which produce the same result as clear bit as explained in Eq. (7), and the weight counters of disagreeing result IP cores are divided by 2 (right shifted) as explained in Eq. (8). It is done to simplify voting circuit hardware implementation. This technique produces supreme results especially when using a minimum number of IP cores.

$$\mathrm{W}|\mathbf{initial} = 0 \tag{4}$$

$$\mathrm{W}|[\mathrm{b}=0] = \sum_{b=0} \mathrm{Wbx} \tag{5}$$

$$\mathrm{W}|[\mathrm{b}=1] = \sum_{b=1} \mathrm{Wbx} \tag{6}$$

$$\mathrm{Wi}+1|\mathrm{Clear} = \mathrm{Wi}+1 \tag{7}$$

$$\mathrm{Wi}+1|\mathrm{Trojan} = \mathrm{Wi}/2 \tag{8}$$

Fig. 5 explains the flowchart of weighted voting technique.

Simulation

A lightweight Java simulator application is developed from scratch using Java Programming Language to simulate triggering Trojans in odd number of IP cores, do both majority voting techniques on the output and generate needed statistics.

Input for simulation process is 4 values:

1. *Trojan Trigger Probability* Array (TTP): it is a fraction number less than 1 to represent the Trojan trigger probability in each IP core.
2. *Number of IPs (Nips)*: It is a positive odd number to represent how many IP cores will be used in our simulation.
3. *IP Word Length (WL)*: it is positive number to represent the word length for the used IP cores.
4. *Total number of Samples (Ns)*: it is a positive integer number to represent the total number of simulation samples.

Output from simulation is the results for both simple and weighted voting such as:

1. Probability of occurrence (PO) = number of actual occurrence Trojans (Not)/{number of IP (Nips) ∗ number of samples (Ns)} as explained in Eq. (9).

$$\mathrm{PO} = \mathrm{Not}/(\mathrm{Ns} * \mathrm{Nips}) \tag{9}$$

2. Probability of detection (PD) = number of detected Trojans (Ndt)/number of generated Trojans (Ngt) as explained in Eq. (10).

$$\text{PD} = \text{Ndt}/\text{Ngt} \tag{10}$$

3. Number of false positives (Nfps): It occurs if the reported IP core has no Trojan.
4. Number of false negatives (Nfpn): It occurs if there is no reported Trojan (alarm) while the final output is infected.
5. Probability of successful attacks (PS) = number infected results (Ntr)/number of generated alarms (Na) as explained in Eq. (11).

$$\text{PS} = \text{Ntr}/\text{Na} \tag{11}$$

Hardware overhead versus security cost

Cost of security is a necessary metric for evaluating any security countermeasure. Being able to determine security costs accurately is a prerequisite for any cost benefit calculation. Our countermeasure consists of multi-vendor hardware duplication with extra voting circuit which adds overhead in terms of hardware such as path delay, consumed power and extra gates area. Verilog is used as HDL (Hardware Description Language) to describe our voting modules in order to simulate and measure the overhead. Details are explained in experiment 3.

Experimental results

Three main experiments have been achieved to measure the effectiveness of our proposed voting techniques and measure the hardware overheads:

Experiment (1)

Simulation is done for many times to conduct a comparison between simple and weighted voting using 3 IP cores (IP1, IP2 and IP3) with all possible combinations of trust. It is done using our Java simulator. Table 2 is generated from simulation, we assume that high trusted IP core (H) has trigger probability of 0%; moderate trusted IP core (M) has trigger probability of 1%; and lower trusted IP core (L) has trigger probability of 10%.

- PDs: Probability of detection for simple voting.
- PDw: Probability of detection for weighted voting.
- Pfps: Probability of false positives for simple voting.
- Pfpw: Probability of false positives for weighted voting.
- Pfns: Probability of false negatives for simple voting.
- Pfnw: Probability of false negatives for weighted voting.

Table 2 shows that weighted voting is showing a better security performance than simple voting especially if there is higher trust in the used IP cores. It is also noticed that weighted voting is producing minimal false negatives than simple voting. Regarding false positives, both of them are almost producing the same results.

Experiment (2)

It measures the suitable number of IP cores on the both of voting techniques that generate detection probability of 100%. Simulation results show that using simple voting among 9 different IP cores is achieving the ideal result as same as using weighted voting among 3 different IP cores as shown in Figs. 6–8.

Detection ratio (Rd) is the ratio between probability of detection for simple voting and probability of detection for weighted voting as shown in Eq. (12). False positives ratio (Rfp) is the ratio between probability of false positives for simple voting and probability of false positives for weighted voting as shown in Eq. (13). False negatives ratio (Rfn) is the ratio between probability of false negatives for simple voting and probability of false negatives for weighted voting as shown in Eq. (14).

$$\text{Rd} = \text{Pds}/\text{Pdw} \tag{12}$$

$$\text{Rfp} = \text{Pfps}/\text{Pfpw} \tag{13}$$

$$\text{Rfn} = \text{Pfns}/\text{Pfnw} \tag{14}$$

Experiment (3)

It measures hardware overhead for both simple and weighted voting circuits. Verilog is used to write the description of both. Synopsis DC (Design Compiler) and PT (Prime Time) are used to synthesis and measure the overhead. Hardware overhead KPIs are based on circuit delay, leaked power and extra added area.

As shown in Fig. 6 that ideal results {100% detection, zero false positives and zero false negatives} are achieved by using only 3 IP cores with weighted voting circuit. Same ideal results are achieved using 9 IP cores with simple voting circuit; our

Table 2 Simple and weighted voting results.

IP1	IP2	IP3	PDs (%)	PDw (%)	Pfps (%)	PFpw (%)	PFns (%)	PFnw (%)
L	L	L	80.8998	59.9439	10.0669	50.0096	0.3664	0.2749
L	L	M	88.8216	94.1887	5.8448	8.4551	0.0533	0.0489
L	L	H	89.9835	100.0000	5.2723	0.0000	0.0000	0.0000
L	M	H	98.1624	100.0000	0.9273	0.0000	0.0000	0.0000
M	M	L	96.4453	90.7573	1.7970	14.7709	0.0085	0.0078
M	M	M	98.0031	65.9921	1.0035	49.9444	0.0034	0.0025
M	M	H	98.9235	100.0000	0.5411	0.0000	0.0000	0.0000
H	H	L	100.0000	100.0000	0.0000	0.0000	0.0000	0.0000
H	H	M	100.0000	100.0000	0.0000	0.0000	0.0000	0.0000
H	H	H	100.0000	100.0000	0.0000	0.0000	0.0000	0.0000

Table 3 ODD-number of IPs results.

IPs#	Detection Ratio (PDs/PDw)	False +ves ratio (Pfps/Pfpw)	False −ves ratio (Pfns/Pfnw)
3	0.9902	0.00493	0.00003
5	0.9998	0.00015	0.00000
7	0.9999	0.00005	0.00000
9	1.0000	0.00000	0.00000
11	1.0000	0.00000	0.00000
13	1.0000	0.00000	0.00000
15	1.0000	0.00000	0.00000

Table 4 Hardware overhead for both ALU and AES.

Voting circuit	Delay overhead		Power overhead		Area overhead	
	Simple	Weighted	Simple	Weighted	Simple	Weighted
Number of IPs	9	3	9	3	9	3
AES	2.90%	39.10%	833.11%	287.35%	494.25%	369.53%
ALU	2.80%	37.80%	219.12%	13.57%	252.31%	234.37%

experiment shows the hardware overhead for both ideal combinations. Experiment 3 is done on two IP cores: ALU and AES. The description of each core is as below:

- ALU IP core: This core presents a "4-bits" ALU. It performs arithmetic operations such as: addition, subtraction, multiplication, and division. It also supports logical operations like AND, OR, XOR, and NOR.

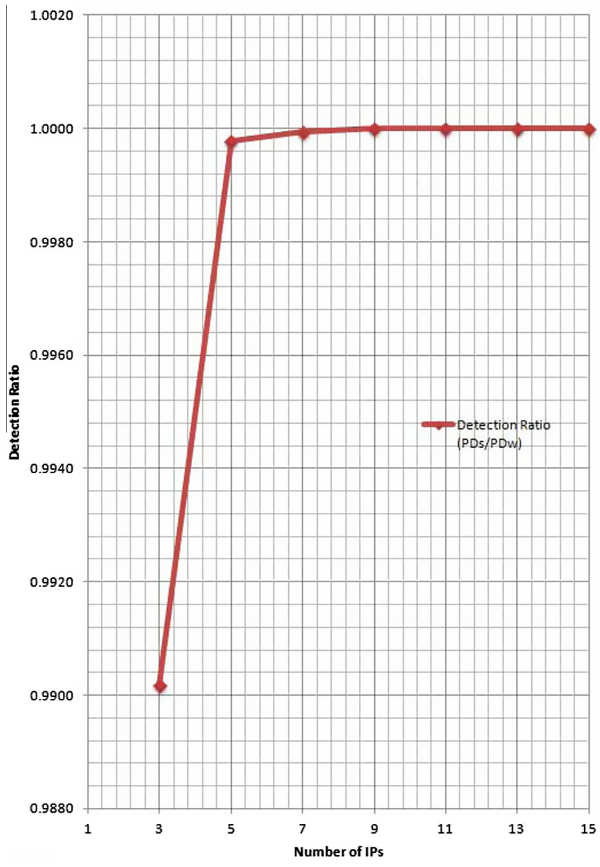


Fig. 6 Detection ratio.

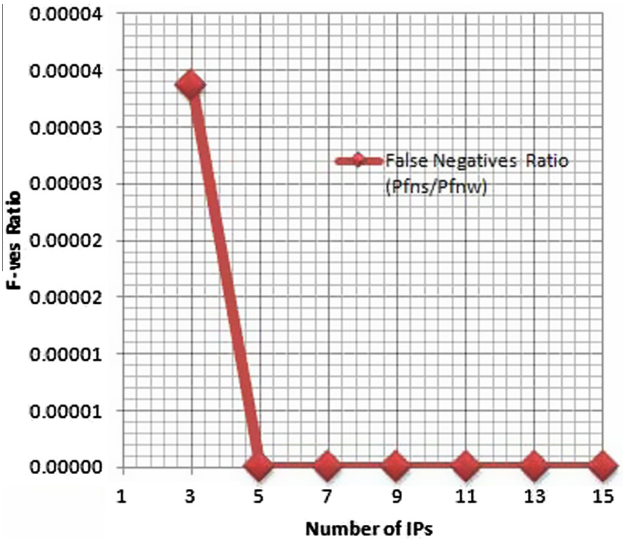


Fig. 7 False negatives ratio.

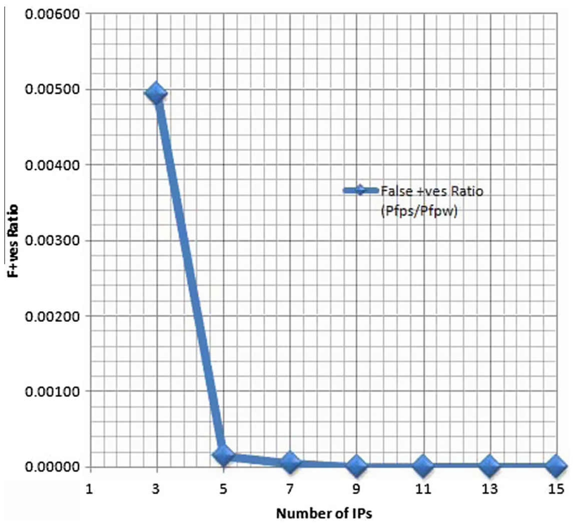


Fig. 8 False positives ratio.

- AES IP core: This core represents an advanced encryption standard (AES) core with 128 bits key. It supports 128 bits plain text port.

Conclusions

We have showed the methodology that aims to dynamically protect from hardware Trojans embedded in third party IP cores during regular operation. The method operates at run-time instead of the traditional test-time techniques. Also, we aim to protect from Trojans without the availability of golden reference IP core. We presented both simple and weighted majority voting among different implementations of the same IP core from different vendors. In contrast to simple voting, weighted voting gives a higher voting weight to more trusted IP cores. Initial trust levels in the IP cores are initially set by the user; however, they are automatically fine-tuned at run time.

We studied security performance and the hardware implementation overhead of both voting methods. Experimental results showed that the weighted voting method using three IP cores variants are almost equivalent to the simple voting using nine IP cores variants. This justified the lower hardware overhead of the weighted voting method despite the higher complexity of the weighted voting circuit over simple voting.

Conflict of interest

The authors have declared no conflict of interest.

References

[1] Waksman A, Sethumadhavan S. Silencing hardware backdoors. In: Proceedings of the 2011 IEEE symposium on security and privacy, SP '11. IEEE Computer Society; 2011. p. 49–63.

[2] Tehranipoor M, Koushanfar F. A survey of hardware Trojan taxonomy and detection. In: IEEE des test, vol. 27; 2010. p. 10–25.

[3] Beaumont M, Hopkins B, Newby T. Hardware Trojans–prevention, detection, countermeasures. DSTO, defense science and technology organization, PO Box 1500, Edinburgh, South Australia 5111, Australia, 2011.

[4] Tehranipoor M, Salmani H, Zhang X, Wang X, Karri R, Rajendran J, et al. Trustworthy hardware: Trojan detection and design for-trust challenges. Computer 2011;44(7):66–74.

[5] Wang X, Salmani H, Tehranipoor M, Plusquellic J. Hardware Trojan detection and isolation using current integration and localized current analysis. In: Proceedings of the IEEE international symposium on defect and fault tolerance of VLSI systems; 2008. p. 87–95.

[6] Wei S, Potkonjak M. Scalable hardware Trojan diagnosis. IEEE transactions on very large scale integration (VLSI) systems, (99); 2011. p. 1–9.

[7] Banga M, Chandrasekar M, Fang L, Hsiao M. Guided test generation for isolation and detection of embedded Trojans in ICs. In: Proceedings of the 18th ACM Great Lakes symposium on VLSI, GLSVLSI '08; 2008. p. 363–6.

[8] Banga M, Hsiao M. A novel sustained vector technique for the detection of hardware Trojans. In: Proceedings of the 22nd international conference on VLSI design; 2009. p. 327–32.

[9] Banga M, Hsiao M. Vitamin: voltage inversion technique to ascertain malicious insertions in ICs. In: Proceedings of the IEEE international workshop on hardware-oriented security and trust, HST '09; 2009. p. 104–7.

[10] Du D, Narasimhan S, Chakraborty R, Bhunia S. Self-referencing: a scalable side-channel approach for hardware Trojan detection. In: Proceedings of the 12th international conference on cryptographic hardware and embedded systems, CHES'10; 2010. p. 173–87.

[11] Jin Y, Makris Y. Hardware Trojan detection using path delay fingerprint. In: Proceedings of the IEEE international workshop on hardware-oriented security and trust; 2008. p. 51–7.

[12] Koushanfar F, Mirhoseini A, Alkabani Y. A unified submodular framework for multimodal IC Trojan detection. In: Proceedings of the 12th international conference on information hiding, IH'10; 2010. p. 17–32.

[13] Potkonjak M, Nahapetian A, Nelson M, Massey T. Hardware Trojan horse detection using gate-level characterization. In: Proceedings of the 46th annual design automation conference, DAC '09; 2009. p. 688–93.

[14] Alkabani Y, Koushanfar F. Consistency-based characterization for IC Trojan detection. In: Proceedings of the 2009 international conference on computer-aided design, ICCAD '09; 2009. p. 123–7.

[15] Rad R, Plusquellic J, Tehranipoor M. A sensitivity analysis of power signal methods for detecting hardware Trojans under real process and environmental conditions. In: IEEE trans very large scale integr syst, vol. 20; 2010. p. 1735–44.

[16] Rajendran J, Jyothi V, Sinanoglu O, Karri R. Design and analysis of ring oscillator based design-for-trust technique. In: 2011 IEEE 29th VLSI test symposium (VTS); 2011. p. 105–10.

[17] Salmani H, Tehranipoor M, Plusquellic J. New design strategy for improving hardware Trojan detection and reducing Trojan activation time. In: Proceedings of the IEEE international workshop on hardware-oriented security and trust, HOST '09; 2009. p. 66–73.

[18] Zhang X, Tehranipoor M. Case study: detecting hardware Trojans in third-party digital IP cores. In: 2011 IEEE international symposium on hardware-oriented security and trust (HOST); 2011. p. 67–70.

[19] McIntyre D, Wolff F, Papachristou C, Bhunia S. Dynamic evaluation of hardware trust. In: IEEE international symposium on hardware-oriented security and trust; 2009. p. 108–11.

[20] Baumgarten A, Tyagi A, Zambreno J. Preventing IC piracy using reconfigurable logic barriers. In: IEEE des test, vol. 27(1); 2010. p. 66–75.

[21] Newgard B, Hoffman C. Using multiple processors in a single reconfigurable fabric for high-assurance applications. In: HOST. IEEE Computer Society; 2010. p. 25–9.

[22] Beaumont M, Hopkins B, Newby T. Safer path: security architecture using fragmented execution and replication for protection against Trojaned hardware. In: DATE. IEEE; 2012. p. 1000–5.