IJCTA, 8(5), 2015, pp. 1687-1693

System-Level Protection(SLP) and Hardware Tro jan detection in 3PIP cores using Voting Techniques

Ravi M.R.*

Abstract: Hardware Trojan is a malicious modification of Integrated Circuits. When underlying hardware is infected with Trojan, it can by-pass security mechanism implemented at higher layer thus sabotaging the entire or part of circuit when deployed in field. Hardware Trojan detection is normally done after manufacturing. The existing Trojan de-tection methodology require golden design to compare Trojan parameter such as timing, power etc. These detection methods consume a lot of time and are costly too. Trojan detection in Third Party Intellectual Property (3PIP) is challenging. In this paper we propose voting circuit for Trojan detection and system-level protection at run-time. This method requires three designs from three different vendors doing same functionality and builds trust slowly from untrusted implementation of Intellectual property (IP) cores. The module with more weights is more trusted and module with less weight is faulty or probably infected with Trojan. In the proposed voting technique, randomized and graded weighted voting techniques are applied. The randomized version learns the weights to detect outliers. The graded version updates the IP weights gradually. The randomized version learns suitable weights very fast to detect Trojan and improves the probability of detection.

Keywords: Hardware Trojan. •Third-Party Intellectual Property (3PIP) •Golden free •On-line learning •Graded weighted voting.

1. INTRODUCTION

Hardware Trojan is a malicious modification of Integrated Circuit (IC) by an adversary either in the design phase or in the manufacturing phase of an IC. Trojan detection gained much importance these days because of outsourcing of design. It is costly to maintain foundry and difficult to do all operations of IC design life cycle by a single designer. Trojans are mostly available as Hardware Description Language (HDL) source code. The adversary may add or remove single gate to sabotage the entire or part of circuit, when deployed in field. It is difficult to detect one gate modification in million lines of code. To design a prototype, IP cores are most widely due to reduction in design/ verification cost and time. Trojan detection in Third Party Intellectual Property(3PIP) is challenging. The 3PIP cores fall in three categories soft, hard, firm IP cores. The soft are described as as a Very high speed integrated circuit hardware description language(VHDL) or verilog available as source code. The hard is described in physical level description available as Graphic database system (GDSII) files. Firm are available as synthesized libraries. Voting techniques can be applied at various level from gate, Register Transfer Logic (RTL) logic design, functional modules, and IP cores, even though at the Integrated Circuit(IC) and macro-level devices. In [3] explained various threats and security solutions available for IP cores. In [7] taxonomy of Hardware Trojans. Further redundancy method i.e., majority voting can be used for Trojan detection. It can detect functional modification and denial of service Trojan. In [9] Formal verification, coverage analysis, redundant circuit removal, sequential automatic test pattern generation (ATPG), and equivalence theorems are used for Trojan detection in 3PIP cores and does not guarantee 100 % detection. In [8] use FIDelity Enhancing Security (FIDES) methodology for FPGAs that uses a combination of access control policies and behavior learning techniques for anomaly detection. In [5] hardware

* TIFAC-CORE in Cyber Security, Amrita Vishwa Vidyapeetham University, Amrita School of Engineering, Coimbatore, India, *Email: mrravi1230@gmail.com*

anchor, which bridges communication between hardware and operating system (OS) to monitor malicious operation due IP core. In [10] explains that rather than detection preventive action can be a better solution. The solutions are power reset, data obfuscation, and sequence breaking. In [6] mentions methods like logic, path delay, current, power, thermal, and hybrids analysis for Trojan detection in IC. These are impractical when circuit size is large.

The effect of Hardware Trojan attack ranges from change in hardware function to degrade in system performance to denial of service. Traditional Trojan detection is done after manufacturing since user do not have control on modification done by adversary. The existing detection methods are costly. Field Programmable Gate Array (FPGA) modeled prototypes are used to design a prototype. These are widely used in military, space, aviation, transport, etc. In this research work, voting based Trojan detection method is used since it does not need design modification. The objective in this work are Trojan detection and system-level protection.

In this paper change functionality, degrade performance, and denial of service (not leakage) Trojan detection are considered. The rest of the paper is organized as follows: Section 2 describes the related works. In Section 3, the proposed system is described. Section 4 illustrates the experimental results and the conclusion is drawn in Section 5.

2. RELATED WORKS

In [2] a weighted voting technique for Trojan detection which is better than simple voting was proposed. The simple voting can detect Trojan when at least one bit of the module is in error at any time. The weighted voting can detect Trojan if at-least one module bit is trusted at any time. The weighted voting technique has less probability of detection if all the module bits are untrusted. In [4] a randomized version of weighted voting which is better than weighted voting was proposed. The randomized version of voting technique was able to learn and identify Trojan infected IP core outlier. In [11] homomorphic data isolation for Trojan Protection that encapsulates IP core with encryption and decryption was proposed. In [1] Trojan detection methodology based on external Cyclic-Redundancy Check (CRC) was proposed.

3. PROPOSED SYSTEM

A block diagram for the proposed system is shown in Figure 1.

Even though IP cores design are implemented by different vendors, the functionality of each IP core is the same. The IP cores from different vendors are assumed that no two Trojans affect the same bit simultaneously. Also the probability of Trojan Trigger is different for different IP cores. When same input is fed to different IP cores, the IP cores have to produce same output. But due to Trojan insertion or fault, the IP cores produce different output. The voter masks the output of the faulty IP cores from reaching the final output. The weighted voter keeps track of IP cores output contribution to final output in terms of weights. These weights are used in the next voting cycle to

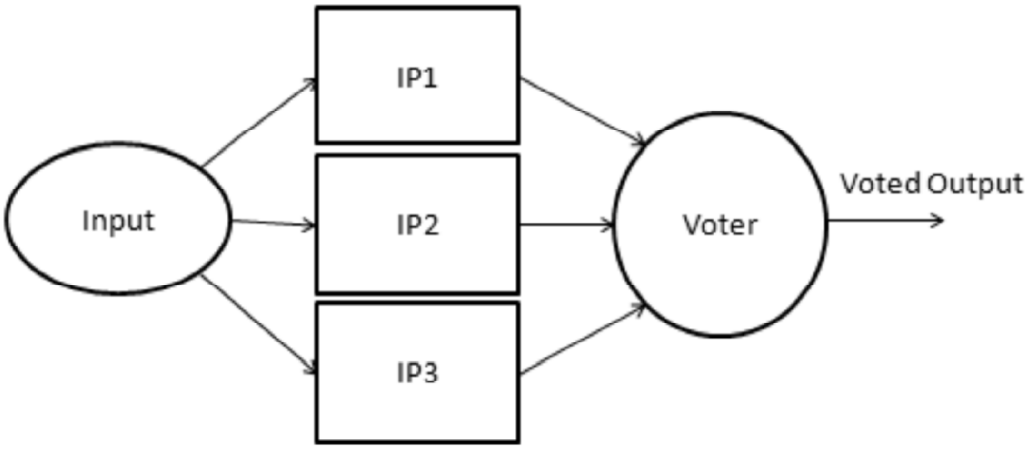

Figure 1: Triple Modular Redundancy(TMR) system with voter for Trojan Detection

produce reliable output. These weights indicate the importance of IP output. The tuned weights indicate whether the Trojan actually activated or not. Voter masks the fault at IP cores output bit from reaching the final output. This provides system-level protection and further improves the reliability of the system.Figure 2 shows the four bit Arithmetic Logic Unit(ALU) along with voters.

The number of voters depends on the number of bits. Here 4 voters are used. *vb*0, *vb*1, *vb*2, and *vb*3 are the output of voters. The same bits from I P 1, I P 2, and I P 3 are fed to the voting circuit. The input for voting circuit *b*0 are from *b*0 of I P 1, *b*0 of I P 2, and *b*0 of I P 3. The inputs for voting circuits *b*1, *b*2, and *b*3 follow similar pattern as in voting circuit for b0.

3.1. Stuck-At-Zero (SAZ) Tro jan

The SAZ Trojan is simply an AND gate. The trigger happens rarely. It is from internal or external circuit and payload affects the output of one of the ALU bit.

Once the Trojan gets activated, the output is always at logic zero. The block diagram of SAZ Trojan in a four bit ALU is shown in Figure 3.

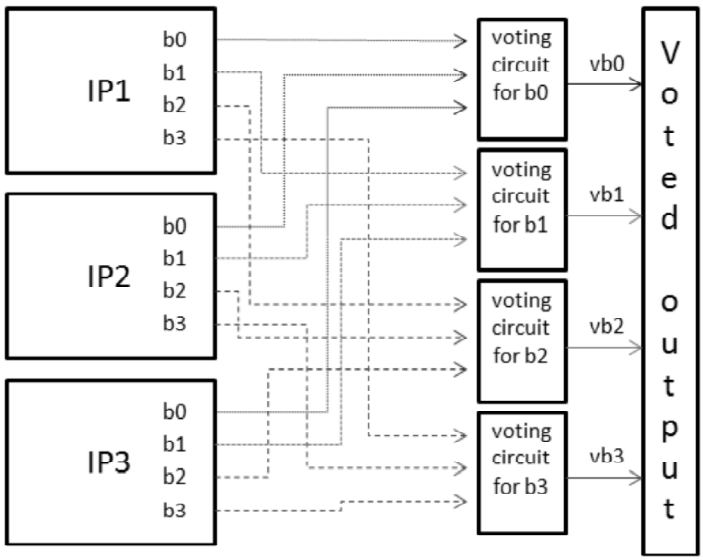

Figure 2: Four bit ALU TMR configuration

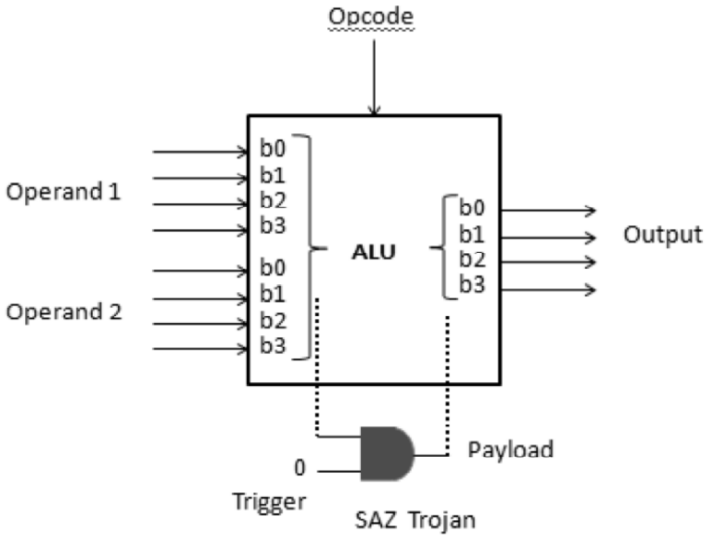

Figure 3: SAZ Trojan

3.2. Tro jan Detection Methodology

The IP core output is given weights. These weights are tuned on-line by voting algorithm to give trusted output. To detect Trojan voting circuit is run up to 104 cycles with different input combination.Final output with higher weights indicate the level of trust. The IP core with higher weights is more trusted and IP core with lower weights is less trusted. At each cycle the Trojan is triggered and checked whether it is detected at the IP core output. It is also checked whether the voter gives trusted output in spite of Trojan activation. The Trojans in IP core are triggered with different probability and ensured that no two Trojans triggered simultaneously i.e always single Trojan is Triggered.

Table 1
Randomized version of weighted voting

IP cores output			*Win*			*Bit selection*	*Wout*			*Probability*	*Voted output*
IP1	*IP2*	*IP3*	*W1*	*W2*	*W3*	*Unirand*	*W1*	*W2*	*W3*	*Po*	*Vb0*
0	0	0	1	1	1	0	1	1	1	3/3	0
0	0	1	2	2	2	0	2	2	2	2/6	0
0	1	0	1	1	3	0	1	1	3	4/5	0
0	1	1	2	0	4	1	3	0	4	4/6	1
1	0	0	1	1	5	0	1	1	5	6/7	0
1	0	1	0	2	6	1	0	2	6	6/8	1
1	1	0	1	1	7	1	1	1	7	2/9	1
1	1	1	2	2	3	1	2	2	3	7/7	1

4. VOTING ALGORITHMS

The proposed voting algorithms for Trojan detection and system-level protection are randomized version of weighted voting and graded penalty/reward weighted voting. The weighted voting metrics are compared with metrics of weighted voting(randomized) and graded voting.

4.1. Randomized version of weighted voting

The algorithm for the randomized version of weighted voting is given in Algorithm 1. A working example of Algorithm 1 is shown in Table 1.

Algorithm 1 weighted voting (randomized version)

[*t*]

1: procedure Input same bit IP core output from IP1, IP2, IP3 as (x1, *x*2, *x*3)
2: Initialize IP core bit weights *W*1, *W*2, *W*3 to 1 corresponding to I P 1, I P 2, I P 3
3: Randomly select any one bit IP core output as Final output yi with probability w_i/W
4: where $W = P\, w_i$
5: Increase W1, W 2, W 3 for IPs (Their Final output and IP core output is same)
6: Decrease (boolean right shift by 1) W 1, W 2, W 3 for IPs (Their Final output and IP core output is different)
7: Update weights W 1, W 2, W 3 used as initial weights in next voting cycle
8: Output randomly selected bit as output of voter with probability P 0
9: end procedure

Initially the bit weights of b0 of I P 1, I P 2, I P 3 is initialized to 1, 1, 1. When the IP core output for b0 of I P 1, I P 2, I P 3 is 0, 0, 0 The output bit is selected randomly using uniform rand function. Any bit of I P 1, I P 2, I P

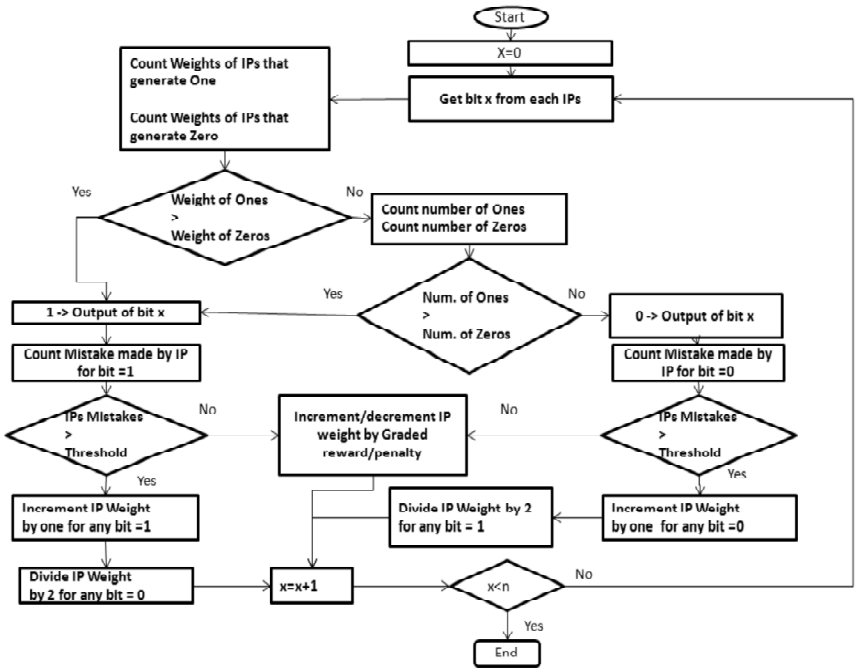

Figure 4: Graded reward/penalty weighted voting

3 may be selected as Final output with probability. The probability is weight input for that bit divided by total weight. Now the final output and IP output are compared to

increase/decrease the weight input and stored in weighted output. The final output and IP output is same weight is increased. The final output and IP output is not same weights are decreased. This weight output is then used in next voting cycle to calculate probability. Thus the learned weights indicate the outliers i.e bit infected with Trojan.

4.2. Graded penalty/reward weighted voting

The working of graded weighted voting algorithm is shown in Figure 4. The reward and penalty are updated based on mistake count. The penalty for IP bit is increased if bit output of IP and corresponding final bit output are different. This value is stored in penalty counter. If IPs bit output and corresponding final bit output are same, it is a reward for IP and stored in reward counter. If reward/penalty is 1,2,3 the weight of IPs bit is incremented/ decremented by 0.25,0.5,0.75 respectively. The learning is retained in weights and these are increased gradually. In weighted voting the weights are suddenly changed from

0 to 1 i.e 0/1 means no contribution/contribution to output. Even though the output of IP core cannot contribute because in previous cycle it gave a fault output. This problem is addressed in graded voting. Here, the mistake threshold is four. when threshold is reached, the graded version works as normal weighted voter.

5. EXPERIMENTAL RESULTS

The ALU is assigned with different level of trust viz., High(H), Medium(M), Low(L), which are as follows:

- H means 0 % Trojan Trigger probability.
- M means 1 % Trojan Trigger probability.
- L means 10 % Trojan Trigger probability.

The ALU's are arranged in different combination of Trust

H H H, LLL, LLM, LLH, LM H, M M L, M M M, M M H, H H L, H H L, H H M. The identical input is given

to each combination and IP cores output is given to different voter under consideration. The final output is taken from output of voter. The metrics for Trojan detection are the following :

1. Probability of detection (PD),
2. Probability of false positive (Pfp),
3. Probability of false negative (Nfn).

The probabilty of detection (PD)= number of detected Trojas (Ndt)/ number of generated Trojans(Ngt).

$$P\ D = N\ dt/N\ gt \quad (1)$$

Number of false positives(Nfp): it occurs when there is no Trojan in IP cores but there an alarm is raised that Trojan is present.

Number of false negatives (Nfn): it occurs where there is no reported Trojan (alarm) but final output is infected.

Probability of detection of weighted voting (PDw)

Probability of detection of graded voting (PDg)

Probability of false positives of weighted voting (Pfpw)

Probability of false positives of graded voting (Pfpg)

Probability of false negative of weighted voting (Pfnw)

Probability of false negative of graded voting (Pfng)

The Trojan detection metrics are tabulated in Table 2

Graded voting technique provides better probability of detection and less false positive and false negative compared to weighted voting in LLL, LLM, M M L, M M M trust configurations.

6. CONCLUSION

In this paper three different voting algorithms such as weighted voting, randomized weighted voting, and graded weighted voting were used for the detection of Hardware Trojan. The MATLAB simulation was carried out with 10 different trust combinations using three IP cores. The probability of detection with simple voting is 100 % when at least two modules of the three IP cores were trusted. The probability of detection with weighted voting is 100 % when at least one module was trusted. The randomized version of weighted voting learned weights faster to detect Trojans. The probability of detection of weighted voting technique was improved with graded voting techniques.

Table 2
Comparing weighted and graded weighted voter

IP cores			*Detection rate*		*FN rate*		*FP rate*	
IP1	*IP2*	*IP3*	*PDw(%)*	*PDg(%)*	*Pfpw*	*Pfpg*	*Pfnw*	*Pfng*
L	L	L	60	83	50	0	.2749	0
L	L	M	94	98	8.4	0	.0489	0
L	L	H	100	100	0	0	0	0
L	M	H	100	100	0	0	0	0
M	M	L	91	98	15	0	.0078	0
M	M	M	66	80	50	0	.0025	0
M	M	H	100	100	0	0	0	0
H	H	L	100	100	0	0	0	0
H	H	M	100	100	0	0	0	0
H	H	H	100	100	0	0	09	0

References

[1] Al-Anwar, A., Alkabani, Y., El-Kharashi, M.W., Bedour, H.: Hardware trojan detection methodology for fpga. In: Communications, Computers and Signal Processing (PACRIM), 2013 IEEE Pacific Rim Conference on. pp. 177–182. IEEE (2013).

[2] Amin, H.A., Alkabani, Y., Selim, G.M.: System-level protection and hardware trojan detection using weighted voting. Journal of Advanced Research 5(4), 499–505 (2014).

[3] Basak, A., Mal-Sarkar, S., Bhunia, S.: Secure and trusted soc: Challenges and emerging solutions. In: Microprocessor Test and Verification (MTV), 2013 14th International Workshop on. pp. 29–34. IEEE (2013).

[4] Blum, A.: On-line algorithms in machine learning. Springer (1998).

[5] Jin, Y., Oliveira, D.: Trustworthy soc architecture with on-demand security policies and hw-sw cooperation. In: 5th Workshop on SoCs, Heterogeneous Architectures and Workloads (SHAW-5) (2014).

[6] Kitsos, P., Voyiatzis, A.G.: Towards a hardware trojan detection methodology. In: 2nd EUROMICRO/IEEE Workshop on Embedded and Cyber-Physica Systems (ECYPS 2014), Budva, Montenegro. pp. 15–19 (2014).

[7] Reece, T.: Assessing and Detecting Malicious Hardware in Integrated Circuits. Ph.D. thesis, Vanderbilt University (2014).

[8] Shila, D.M., Venugopalan, V., Patterson, C.D.: Fides: Enhancing trust in recon-figurable based hardware systems. Cryptology ePrint Archive, Report 2015/441 (2015), http://eprint.iacr.org/.

[9] Tehranipoor, M., Salmani, H., Zhang, X.: Hardware trojan detection: Untrusted third-party ip cores. In: Integrated Circuit Authentication, pp. 19–30. Springer (2014).

[10] Waksman, A., Sethumadhavan, S.: Silencing hardware backdoors. In: Security and Privacy (SP), 2011 IEEE Symposium on. pp. 49–63. IEEE (2011).

[11] Ziad, M., Alanwar, A., Alkabani, Y., El-Kharashi, M.W., Bedour, H.: Homomorphic data isolation for hardware trojan protection. arXiv preprint arXiv:1505.05226 (2015).