Hardware Trojan Detection Methodology for FPGA

Amr Al-Anwar, Yousra Alkabani, M. Watheq El-Kharashi, Hassan Bedour
Department of Computer and Systems Engineering
Ain Shams University
Cairo 11517, Egypt
{amr.alanwar, yousra.alkabani, watheq.elkharashi, hassan.bedour}@eng.asu.edu.eg

Abstract—**Nowadays, hardware Trojan protection became a hot topic especially after the horizontal silicon industry business model. Third party IPs are the building blocks of many critical systems and that arises the question of confidentiality and reliability of these blocks. In this work, we present novel methods for system protection and Trojan detection that alleviate the need for a golden chip. In addition, we introduce a scenario to dynamically remove infected IPs embedded on FPGAs. We propose multiplexing reconfigurable IPs' outputs and a CRC Trojan detection schema to detect Hardware Trojan. Dynamic Trojan detection is done using multiple variant voting. We show the practicality of the introduced schemes by providing a proof of concept implementation of the different methodologies on FPGAs. We investigate methods' overhead to provide superior security properties that can be used in critical and strategically important systems.**

I. Introduction

One of the most important problems to resolve nowadays is hardware Trojan circuits. End-users need to get guarantees that their devices are reliable enough, are not controlled by unknown entities, and will not leak sensitive information. Designers need to have guarantees that their designs are not tampered with while maintaining technology secrets of the fabrication facilities and design royalties of third party IP owners. Hardware Trojan appears to be one of the most critical topics with the widespread use of silicon chips in different applications, varying from cell phones, cars, to strategically important military devices. It is important to provide methods that resolve the trust issues between fabrication facilities, designers, and end-users.

Most design houses are currently fabless and tend to manufacture their designs in offshore facilities. That is due to the increase in systems complexity, the desire to have shorter time to market, increasing cost of maintaining manufacturing facilities, and the increase in the competition. Moreover, designers tend to embed third party IPs in their designs. This raises many trust issues in the design industry in many directions. Designers need to ensure that no malicious circuitry is embedded in their designs by fabrication facilities or third party IP owners. At the same time, the fabrication process is obscured from the designer and the details of third party IPs are hidden to protect the IP owner's rights.

Hardware Trojan detection methods have been developed to ensure that chips are Trojan-free. These methods either try to detect the existence of a Trojan with high probability by studying side channels [2], [3], [7]–[10], or try to introduce architectural modifications to prevent embedding Tojan task [4], [11]–[13]. However, these methods mainly depend on comparing the suspected chip with a golden one (a known non-infected chip). In practice, a golden chip might not always be readily available, especially when using third party IPs. Attempts to depend on using the system integrator's design specifications for comparisons were introduced [19].

In this paper, we develop novel methods for Trojan detection and protection. We introduce two methods that provide high levels of protection. We can divide the presented work into: (1) Multiplexing reconfigurable IPs outputs and CRC Trojan detection (MCRC), and (2) Multiple variants implementation (MV).

Multiplexing reconfigurable IPs outputs and CRC Trojan detection schema (MCRC) method aims to deal with infected Trojan by decreasing its probability to leak sensitive information until detecting the infected IP and totally removing it by using the partial reconfiguration technology. The multiple variants method (MV) introduces redundancy by using an odd number (at least three) of the same IP from different vendors and voting over the output. While these methods introduce significant larger overhead, they provides higher levels of protection. For instance, they can be used to protect from failure and detect Trojans.

The rest of the paper is organized as follows. Section II summarizes the related work. Trojan detection methodologies are described in section III, which consist of two subsections: 1) Multiplexing Reconfigurable IPs outputs and CRC Trojan detection schema, and 2) Multiple variant implementation, which are described in Subsections III-B, and III-A, respectively. Attacks and countermeasures are provided in Section IV. Experimental evaluation and estimation of the overhead of the methods are shown in Section V. Section VI concludes the work.

II. Related work

Architectural methodologies try to increase the chances of the activation of a hardware Trojan during testing. Salmani et. al increase the Trojan activity by inserting dummy flip flops in the design [13]. They choose the locations of the inserted flip flops based on a transition probability threshold. Rajendran et al. introduced a methodology for securing all the gates of the design using ring oscillators [12]. They add extra logic that converts paths of the circuit into ring oscillators. Changes in the frequency of the ring oscillators are used to detect the presence of Trojans. Banga and Hsiao used voltage inversion at alternating levels of the circuit to increase the power consumption of an infected circuit [4].

Side channel dependent methodologies try to localize the impact of the Trojan on the circuit without activating it. The main idea is to try to detect the presence of a Trojan with high probability by detecting the overload of the Trojan circuit on different circuit parameters; such as the power or the delay as compared to a non-infected circuit. Jin and Makris use path delay analysis to detect Trojans [8]. Wang et. al use localized current analysis for Trojan detection. They measure power from multiple ports to detect the impact of the Trojan on power [17]. The impact of a Trojan on the power supply transient current of an IC is studied using statistical methods by Rad et al. [11]. Banga et al. introduced a test vector generation method that can be used to differentiate between the side-channel waveforms of a Trojan infected and a non-infected circuit [2], [3]. Gate-level characterization techniques accompanied by statistical methods are also used to detect hardware Trojans. These methods depend on gate-level delay or power characterization, or both [1], [9], [10], [18].

Regular testing methods are not suitable to detect the faults caused by hardware Trojans because they are not expected to be activated during testing. This is because Trojan circuits are usually designed to be activated using a rare trigger. Recently, different methods have been specifically designed with Trojan detection in mind. We can classify these hardware Trojan detection methodologies into: (1) side-channel dependent methodologies and (2) architectural methodologies [14].

The main problem of all these methods is that they require the presence of a non-infected (golden) chip. Thus, they are very practical if the design does not contain third party IPs [15]. However, if the system designer integrates third party IPs in his design, these methods become less practical. Zhang and Tehranipoor try to provide an alternative to using a golden design by using formal verification, code coverage analysis, and ATPG methods to achieve high confidence in whether the circuit is Trojan-free or Trojan-inserted [19]. Baumgarten et al. 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 [5]. A. Waksman and Sethumadhavan introduced a method that attempts to prevent Trojan triggering [16]. Beaumont et al. ran replica of a program on multiple processing elements to achieve protection form hardware Trojans [6].

III. Trojan Detection methods

In this section, we introduce two Trojan detection and protection methods and a scenario to remove infected IPs during run time. This section is divided in two subsections: A) Multiplexing Reconfigurable IPs' outputs and CRC Trojan detection schema (MCRC) in Subsection III-A, and B) Multiple variant implementation (MV) in Subsection III-B.

A. Multiplexing Reconfigurable IPs' outputs and CRC Trojan detection schema (MCRC)

We try to increase the defense against Trojan attacks by reducing the probability of both leakage of sensitive information and malfunction due to a Trojan attack. We attempt to do this by depending on two or more untrusted implementations of the same IP from different untrusted vendors. Thus, the actual output of the system will be a mixture of the output from all the different copies in the system. In this methodology, we suggest multiplexing m IPs output. Our enhancement aims not only to decrease the probability of leaking sensitive information by multiplexing IPs' outputs, but also mark the infected IP and its vendor as an attacker. We introduce voting among IPs' CRCs to choose the major CRC, i.e, the CRC for the non infected-data. As shown in Figure 1, the CRC voting circuit will be responsible for that job. Furthermore, the CRC voting circuit will keep track of the number of data errors for each IP. And if this number exceeds a certain threshold, the IP will be marked as an infected one and the alarm circuit will trigger a warning. We provide a small warning score for IPs that show discrepancies for the first time as this might be a design bug not a Trojan. However, this warning increases when more faults are detected. The CRC voting circuit is shown in Figure 2. It is worthy to mention that we can compare IPs output data instead of comparing CRC. But, comparing IPs outputs will cost more area, power and delay overheads as number of CRC output bits are smaller in size than IP output. We suggest not to mark IP as infected at its first CRC error, i.e, choose a threshold value more than one in order to avoid rare case design bugs.

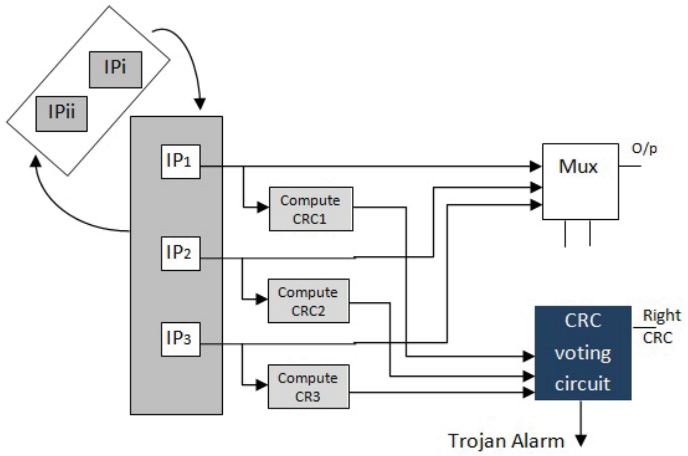

Fig. 1. Multiplexing between Reconfigurable IPs' outputs.

To further decrease the probability of losing information, we enhance the prior idea by using FPGA partial reconfiguration (PR) feature to provide more secure designs on FPGA. Partial reconfiguration divides FPGA logic design into two different types: reconfigurable logic and static logic. It allows modifying reconfigurable regions in the FPGA without compromising the integrity of the applications running on the static logic part, as illustrated in Figure 3. Each IP core will reserve a

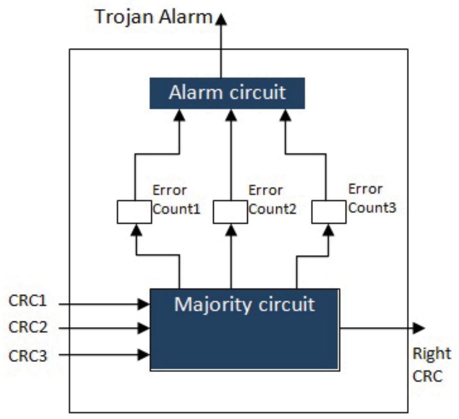

Fig. 2. CRC voting circuit.

reconfigurable logic region, while the mux or even the CRC voting circuit will reserve a static logic region. In order to decrease the chance of sending secret information, we suggest to partial reconfigure the FPGA to replace the current running IP with new IP periodically. We will need a queue of IPs some of them are running currently and others are waiting their turn in the queue as shown in Figure 1. It is interesting to relate the error count of such IP with its running time.

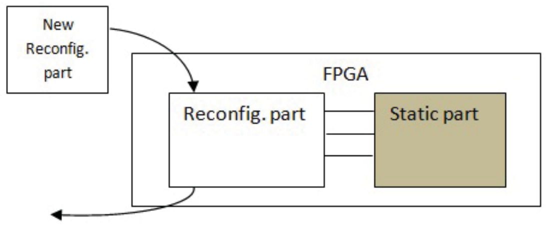

Fig. 3. FPGA partial reconfiguration concept.

We chose cyclic redundancy check (CRC) among other error detection schema as it is the most effective scheme that is commonly used. Its performance is better than other error check algorithms. Furthermore, CRC is able to check multiple bit errors. CRC has strong error detection capability. It could capture 99.95% of the transmission errors. If a single bit is incorrect, the CRC value will not match up.

Also, if the used IPs are network communication IPs, it's more secure to take the packet data from different IP cores, not to take every packet from one of the available IPs. Also, we advise to use built-in CRC computation circuits as attacker may design his IP to provide correct CRC even if the Trojan is active and we can generalize that to any system that use CRC computation. Nevertheless, If an infected IP succeeds to inject minor part of secret information on its turn, specific IP error count should be incremented and receiver would reply with a resend request as CRC computation does not match the received data. So, we can use receiver retransmission request as another sign for being infected.

We should note that the minimum number of needed IPs for using CRC methodology is three IPs as voting is done between CRCs.

Algorithm 1 describes multiplexing between reconfigurable IPs' outputs algorithm for third party cores used on FPGA. The system owner keeps m different implementations of the suspected core from different vendors. The FPGA is configured only using 3 cores, a multiplexer, and a CRC voting circuit. The system runs normally and the output is taken by multiplexing outputs of the different variants. Once the CRC voting circuit detects an error in an IP, its error count will be incremented. Also, if it exceeds the chosen threshold it will be marked as an infected IP and replaced with a new core. Anomalous cores will be out of our system.

Algorithm 1 Multiplexing Reconfigurable IPs' outputs and CRC Trojan detection schema

```
Download 3 variant cores;
Run system normally
  i = 0;
  while running normally do
    if CRC error then
      increment infected IP error counter;
      if IP error counter exceeds threshold then
        partial reconfigure FPGA to remove that IP totally
        and replace it with the first IP on the queue;
        mark the anomalous core as an infected core;
      end if
    end if
    if periodic time elapsed then
      use partial configuration to replace IP number i with
      the first IP on the queue;
      i = (i + 1) mod 3;
    end if
  end while
```

Furthermore, we propose an enhancement in MCRC which will help in automatic dealing with infected third parity IP which can be achieved by the methodology provided at Figure 4. Multiplexer selection lines can be a shift register with a saved key. This key will determine the chosen IP to run at each turn. We propose a full system to ignore infected IP by decreasing its sharing chance in multiplexed output or may fully remove its turn. CRC voting circuit will count each IP error using CRC voting as described before. Updating sequence circuit will use the provided error counts to set each IP share in the final multiplexed output. The more IP error count, the less contribution in the updated selection lines sequence. Also, if the IP error count exceeds specified error threshold, updating sequence circuit will remove IP selection line value from the updated selection lines sequence, i.e, this IP will be totally ignored. This enhancement will be best for ASIC or FPGA with no partial reconfiguration feature. Since it can deal with infected IP without removing it.

$$IPi\_error\_prob = \frac{IPi\_error\_count}{Total\_number\_of\_IPs\_errors} \quad (1)$$

$$IP\_share = IP\_error\_prob * seq\_L * IP\_f \tag{2}$$

where

- IP_share: number of occurrences of IP selection line value among the new sequence values;
- Total_number_of_IPs_errors: summation of all IPs errors
- IP_f: IP threshold flag;
- seq_L: sequence length;

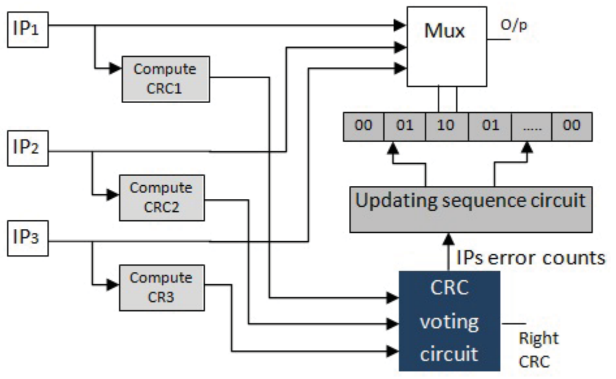

Fig. 4. Enhancement in MCRC schema.

B. Multiple variant implementation (MV)

In this section, we show how to provide superior protection from hardware Trojans by using multiple implementations of the suspected IPs. We propose adding multiple copies of the IP cores from different vendors. Then, using the dynamic Trojan detector circuit to determine the suspected IP and the safe output using a majority voting circuit. Figure 5 shows how the scheme works. We compare the output of m cores and if there is a mismatch between the different outputs, then we suspect that there is a Trojan. If we have three or more IPs we can use a voting circuit that decides the correct output and raise a Trojan alarm, if needed.

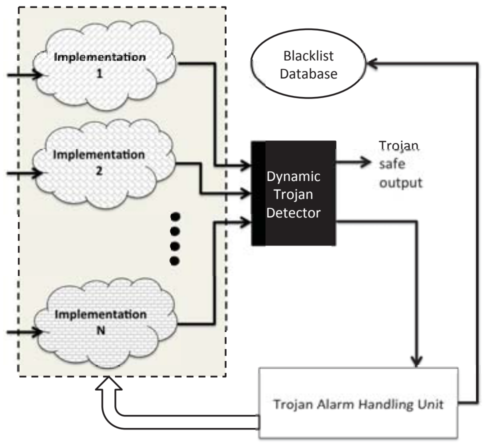

Fig. 5. Using multiple implementations of the same IP to catch Trojan.

If we are using an FPGA, we suggest using partial reconfiguration feature. In our methodology each IP core reserves a reconfigurable logic region, whereas the dynamic Trojan detector unit and the Trojan alarm handling unit will reserve a static logic region. The Trojan alarm handling unit can partially reconfigure the FPGA to remove the infected core and replace it with a new one while the rest of the FPGA logic is still running. The dynamic Trojan detector unit will also mark the anomalous core vendor as a suspected vendor.

Algorithm 2 describes the multiple variant implementation algorithm for third party cores used on FPGA. The system owner keeps m different implementations of the suspected core from different vendors. The FPGA is configured only using 3 cores and the Trojan Alarm circuit. The system runs normally and the output is taken by majority voting from the different variants. Once a Trojan alarm occurs, the anomalous core is declared. The core is replaced by another unsuspected core and is black listed in a cores' Database.

Algorithm 2 Multiple variant implementation algorithm

```
Download 3 variant cores;
run system normally
  while running normally do
    compare IPs outputs and choose major one.
    if one IP output is different than others then
      raise Trojan alarm
      use FPGA partial reconfiguration to replace infected
      core;
      mark the anomalous core as a suspected core in the
      cores' blacklist DataBase;
    end if
  end while
```

We would like here to point out the main difference between this method and the work introduced by Beaumont et al. [6]. Beaumont et al. use multiple processing elements running replica of the code. However, our work applies to any type of hardware IP. In addition, the process described by Beaumont et al. does not provide means for eliminating the hardware Trojan from the circuit, while we describe a scheme of using partial dynamic configuration on FPGAs to eliminate the infected FPGA from the system.

IV. Attacks and Countermeasures

Multiplexing Reconfigurable IPs outputs and CRC Trojan detection schema and Multiple variant implementation can deal with internally triggered and externally triggered Trojans. We don't care about the source of the Trojan Trigger in our methodologies . However, if we analyze more the MCRC schema, we will find that there is a period (T) before the detection of the infected IP. During this period system output is shared between all used IPs. And, if we assume a fair multiplexing between IPs, we will find that the attacker IP will have (T/n) period to leak sensitive information. On the other hand, the MV schema will not allow any period to the attacker, as it will choose the major IPs' output during the running time. We may remark the former argument as an advantage for MV over MCRC. This advantage will take place when addressing leaking information and disturbing functionality Trojans.

Concerning Timing and power side channel attacks, we think that it will be quite difficult for the attacker to deal with our methodologies due to different implementation of the IP and extra circuit for voting or multiplexing.

We can summarize the above information in Table I MCRC denotes Multiplexing Reconfigurable IPs outputs and CRC Trojan detection schema and MV denotes multiple variant implementation. Each methodology will have a score against each attack. A minimum score of 0 means the method cannot protect against the attack. Scores of 1 and 2 refer to moderate and strong protection against the attack, respectively.

TABLE I
TROJANS AND PROPOSED METHODS SCORE.

Trojan type	MCRC	MV
Externally triggred	1	2
Internally triggred	1	2
Leaking info.	1	2
Disturbing functionality	1	2
Power side channel	2	2
Timing side channel	2	2
Method score	8	12

V. EXPERIMENTAL RESULTS

In this section, we study the effect of using the proposed methods on the consumed power, number of used LUT and critical path delay. We will describe in details the used benchmarks and study the overhead of each method. We are going to highlight first the overheads of multiplexing reconfigurable IPs outputs with CRC Trojan detection schema then multiple variant implementation. Before going to numbers and results, we shall define first consumed power as shown in equation 3.

$$P_{consumed} = P_{total} - P_{leakage} \tag{3}$$

1) Multiplexing Reconfigurable IPs' outputs with CRC Trojan detection schema (MCRC): We are going to show results on using multiplexing between IPs' outputs methodology. We used three UART IPs as multiple variants. CRC computation will be done on each UART output. We have to note that using CRC in this case may not be considered as a type of overhead as UART is usually used with CRC computation to detect sending errors. CRC truncated polynomial is $1+x^2+x^5$ and it will be applied on each eight bits UART output. Table II summarizes the synthesis details using Virtex 6 xc6vcx75tff484-1. The first column shows the names of the circuits. The second column shows the number of LUTs used by the circuits. The third and fourth columns show critical path delay and the consumed power of the different UART implementations in addition to CRC circuit. Then, we repeat the prior experiment with ALU IPs. Table III summarizes results. The different ALUs delay, power and area calculations are reported on Virtex 6 xc6vcx75tff484-1. Leakage power equals 1.293W and it is the same for all rows.

TABLE II
MULTIPLEXING UART IPS AND CRC TROJAN DETECTION.

	used LUTs	delay (ns)	P_consumed (W)
UART1	26	1.874	0.013
UART2	28	1.657	0.006
UART3	68	2.371	0.01
FinalUART	179	2.651	0.107

TABLE III
MULTIPLEXING ALU IPS AND CRC TROJAN DETECTION.

	used LUTs	delay (ns)	P_consumed (W)
ALU1	48	12.22	0.002
ALU2	36	4.623	0.013
ALU3	169	3.767	0.05
FinalALU	221	12.528	0.101

2) Multiple variant implementation (MV): Table IV shows the multiple variant experiment results for the UART benchmarks. We selected three different implementations of the UART (denoted UART1, UART2, and UART3) and integrated them with the dynamic Trojan detector circuit to construct FinalUART. FinalUART delay equals max UART modules delay plus the dynamic Trojan detector circuit delay. But in this case, the latter delay is less than 1 ps due to low processing time for the one bit output. We used also Virtex 6 xc6vcx75tff484-1 in this experiment. The first column shows the name of the circuit. The second column shows the number of LUTs used by the circuit. Delay and consumed power are shown in the third and fourth columns, respectively. Leakage power equals 1.293W and it is the same for all rows.

TABLE IV
UART MULTIPLE VARIANT.

	used LUTs	delay (ns)	P_consumed (W)
UART1	26	1.874	0.013
UART2	28	1.657	0.006
UART3	68	2.371	0.01
FinalUART	123	2.371	0.022

Next, we study the overhead of our methodology on multiple cores of ALU IPs. Table V shows the multiple variant experiment results for the ALU benchmarks. We used three different implementations of the ALU denoted by ALU1, ALU2, and ALU3. The overall design including the three different ALU implementations and the dynamic Trojan detector circuit is named FinalALU. The different ALUs delay, power and area calculations are reported on Virtex 6 xc6vcx75tff484-1. Leakage power equals 1.293W and it is the same for all rows. FinalALU critical path delay is 36.988 ns, which is the maximum of the delays of ALU1, ALU2, and ALU3 in addition to the dynamic Trojan detector circuit delay. So, in our case the dynamic Trojan detector circuit delay equals 1.695 ns.

TABLE V
ALU MULTIPLE VARIANT.

	used LUTs	delay (ns)	P_consumed (W)
ALU1	48	12.22	0.002
ALU2	36	4.623	0.013
ALU3	169	3.767	0.05
FinalALU	190	13.372	0.1

AES core is a symmetric block cipher core that can process data blocks of 128 bits, using a 128-bit key. The three different implementations are denoted AES1, AES2, and AES3. FinalAES is the overall circuit including the three AES implementations and the dynamic Trojan detector circuit. Table VI summarizes the synthesis details using Virtex 6 xc6vlx550tff1760-1. The first column shows the names of the circuits. The second column shows the number of LUTs used by the circuits. The third and fourth columns show critical path delay and the consumed power of the different circuits. The dynamic Trojan detector circuit introduces 0.2 ns delay only. Leakage power equals 3.769W and it is the same for all rows.

TABLE VI
AES MULTIPLE VARIANT BENCHMARKS.

	used LUTs	delay ns	P_consumed W
AES1	702	4.313	0.461
AES2	1553	5.502	0.13
AES3	256	2.972	0.419
FinalAES	1153	5.716	1.05

VI. CONCLUSION

We have introduced the first methodologies that work by embracing Trojans and protecting the system from them without trying to detect the Trojan during testing. The methods operate at runtime instead of the traditional test-time techniques. This work targets the protection from Trojans that could be embedded in third party IPs where no golden design is available. We presented methods for detecting the Trojan and removing it from the system. Multiplexing Reconfigurable IPs outputs with CRC Trojan detection schema (MCRC) aims to multiplex IPs' output to decrease the probability of leaking sensitive information. It tries to detect Trojan by comparing CRC of IPs' outputs. Furthermore, we presented an enhancement to the prior method by automatic dealing with the infected IP. We suggested decreasing the sharing chance of IP in the system ouputs upon IP number of errors statistics. Furthermore, we suggest multiple variate schema to chose the major IPs output and suspect unmatched IP output. Both the multiplexing with CRC and the multiple variants can be used to dynamically remove Trojan infected IPs and report Trojans. They take advantage of the partial reconfiguration feature of modern FPGA to clean up the infected IPs.

We provided sample implementation of the different methodologies. To the best of our knowledge, this is the first work that attempts to do dynamic hardware Trojan detection and protection on FPGA.

REFERENCES

[1] Y. Alkabani and F. Koushanfar. Consistency-based characterization for IC trojan detection. In *Proceedings of the 2009 International Conference on Computer-Aided Design*, ICCAD '09, pages 123–127, 2009.

[2] M. Banga, M. Chandrasekar, L. Fang, and M. Hsiao. 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, pages 363–366, 2008.

[3] M. Banga and M. S. Hsiao. A novel sustained vector technique for the detection of hardware trojans. In *Proceedings of the 22nd International Conference on VLSI Design*, pages 327–332, 2009.

[4] M. Banga and M. S. Hsiao. 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, pages 104–107, 2009.

[5] A. Baumgarten, A. Tyagi, and J. Zambreno. Preventing IC piracy using reconfigurable logic barriers. *IEEE Des. Test*, 27(1):66–75, Jan. 2010.

[6] M. Beaumont, B. Hopkins, and T. Newby. Safer path: Security architecture using fragmented execution and replication for protection against trojaned hardware. In *DATE*, pages 1000–1005. IEEE, 2012.

[7] D. Du, S. Narasimhan, R. Chakraborty, and S. Bhunia. 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, pages 173–187, 2010.

[8] Y. Jin and Y. Makris. Hardware trojan detection using path delay fingerprint. In *Proceedings of the IEEE International Workshop on Hardware-Oriented Security and Trust*, pages 51–57, 2008.

[9] F. Koushanfar, A. Mirhoseini, and Y. Alkabani. A unified submodular framework for multimodal IC trojan detection. In *Proceedings of the 12th international conference on Information hiding*, IH'10, pages 17–32, 2010.

[10] M. Potkonjak, A. Nahapetian, M. Nelson, and T. Massey. Hardware trojan horse detection using gate-level characterization. In *Proceedings of the 46th Annual Design Automation Conference*, DAC '09, pages 688–693, 2009.

[11] R. Rad, J. Plusquellic, and M. Tehranipoor. A sensitivity analysis of power signal methods for detecting hardware trojans under real process and environmental conditions. *IEEE Trans. Very Large Scale Integr. Syst.*, 18:1735–1744, December 2010.

[12] J. Rajendran, V. Jyothi, O. Sinanoglu, and R. Karri. Design and analysis of ring oscillator based design-for-trust technique. In *VLSI Test Symposium (VTS), 2011 IEEE 29th*, pages 105 –110, may 2011.

[13] H. Salmani, M. Tehranipoor, and J. Plusquellic. 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, pages 66–73, 2009.

[14] M. Tehranipoor and F. Koushanfar. A survey of hardware trojan taxonomy and detection. *IEEE Des. Test*, 27:10–25, January 2010.

[15] M. Tehranipoor, H. Salmani, X. Zhang, X. Wang, R. Karri, J. Rajendran, and K. Rosenfeld. Trustworthy hardware: Trojan detection and design-for-trust challenges. *Computer*, 44(7):66 –74, July 2011.

[16] A. Waksman and S. Sethumadhavan. Silencing hardware backdoors. In *Proceedings of the 2011 IEEE Symposium on Security and Privacy*, SP '11, pages 49–63. IEEE Computer Society, 2011.

[17] X. Wang, H. Salmani, M. Tehranipoor, and J. Plusquellic. 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*, pages 87–95, 2008.

[18] S. Wei and M. Potkonjak. Scalable hardware trojan diagnosis. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, PP(99):1 –9, 2011.

[19] X. Zhang and M. Tehranipoor. Case study: Detecting hardware trojans in third-party digital ip cores. In *2011 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pages 67 –70, June 2011.