Smart Employment of Circuit Redundancy to Effectively Counter Trojans (SECRET) in Third-Party IP Cores

Mohammed M. Farag, Mohammad A. Ewais
Department of Electrical Engineering, Faculty of Engineering, Alexandria University
Emails: mmorsy@alexu.edu.eg, m.a.ewais@ieee.org

Abstract—**Hardware Trojan horses (HTHs) are malicious inclusions or alterations to hardware designs developed and supplied by untrusted parties. The emerging threat of HTHs has a direct impact on the FPGA design community which mainly relies on third-party IP (3PIP) cores and design reuse practices. Efficient design and detection of HTHs have been the main interest of most related research work, but countermeasures against HTHs have not attained sufficient attention. We advance a novel approach promoting Smart Employment of Circuit Redundancy to Effectively Counter Trojans (SECRET) in 3PIP cores employed in reconfigurable hardware designs. Two identical instances of the protected IP core are employed for observation and operating purposes and a time shift is created between the two core inputs. Trojan detection circuitry is inserted during the design-time to monitor the observation core at run-time. Once a Trojan is detected in the observation core, the operating core with the delayed input is suspended or the identified triggering inputs are isolated for a specific period of time to bypass the Trojan activating trigger. We present the SECRET high-level architecture, a proof-of-concept application to a 3PIP crypto core containing an HTH of our design. The prototype is designed and validated on a Spartan-3 FPGA. Simulation and implementation results show the SECRET feasibility and effectiveness.**

Index Terms—**Hardware Trojan, Run-time Countermeasure, Third-Party IP Core, Reconfigurable Hardware.**

I. Introduction

Moore's law has been enabling further and further reductions in transistor size leading to increasingly complex and expensive designs with longer design cycles. To reduce time to market, ease and facilitate hardware design practices, digital circuit designers and manufacturers tend to rely on third-party IP (3PIP) cores produced by foreign developers. The size of the semiconductor IP market is estimated by two billion Dollars and is expected to reach three Billions by 2016 according to the Gartner research firm [1]. The impact of the hardware IP market and the reliance on it in security-critical applications have raised the issue of untrust in hardware IP cores. An untrusted party might alter the chip design or include Trojans and backdoors during various phases of the IP core production and delivering posing a major security threat.

A Hardware Trojan Horse (HTH) is a malicious insertion or manipulation in a hardwired IC or a soft IP core that aims at causing non-typical functionality, not stipulated by the design specifications. An HTH can change the typical functionality of a chip either by adding more logic, manipulating existing logic and wires, or modifying the chip physical specifications such as interconnect routes or transistor geometries. Some Trojans are always-active, capable of causing harm at any time, while others are usually-off and triggered by rarely-occurring conditions, e.g. a sensor output, a counter value, an internal logic state or a sequence of states, a particular input pattern, or after passing a specific period of operation as in time bombs. HTH effects or payloads include performance degradation, partial or full functionality change, information leakage, or denial-of-service [2].

Efficient design and detection of HTHs have been the subject of extensive research during the last decade. Several solutions have been proposed to detect and identify Trojans, most of them are applied in the pre-deployment phase [3]. Design-time Trojan detection methods include physical inspection, side-channel analysis, functional testing, and Trojan activation methods. Physical inspection methods destructively extract the circuit structure and compare it to a reference structure. Side-channel analysis techniques attempt to detect HTHs by measuring the induced changes of side-channel signals including power and delay. Functional testing stimulates the input of a chip and monitors the output to detect non-typical output values and patterns that might be an indication of Trojan existence. Trojan activation methods aim to increase the likelihood of activating HTHs by generating test vectors and patterns stimulating rarely-occurring triggering conditions during the test and verification phases. Unfortunately, detecting HTHs using compile-time approaches is extremely difficult for several reasons, including the need for Trojan-free golden references, the complexity and large size of modern ICs, the small size and side effect and the hidden nature of HTHs.

Design-for-Trust and Security (DFTS) is the alternative approach to improve Trojan detection by adding security components at the design phase to monitor and protect ICs and IP cores at run-time [4]. Security monitors and wrappers are implemented from the system's security specifications and integrated into the protected IC to detect HTHs at run-time. DFTS protection schemes do not assume the existence of a golden reference or the ability to inspect the internals of the hardware modules. Run-time Trojan detection methods can be efficiently applied to protect 3PIP cores deployed in reconfigurable hardware designs because these platforms can easily host the security components and the design flow can be regularly modified to accommodate the DFTS requirements.

Although many contributions have been presented to both design- and run-time Trojan detection methods, complementary research issues such as recovery from and countermeasures against HTHs are not sufficiently addressed in the literature. In this paper, we propose a DFTS method to counter HTHs in 3PIP cores deployed in reconfigurable hardware platforms such as FPGAs. We promote Smart Employment of Circuit Redundancy to Effectively Counter Trojans (SECRET) in 3PIP cores. Two identical instances of the protected 3PIP core are employed for observation and operating purposes, and a predetermined controllable time shift or delay is created between the two core inputs. The IP core input and its time-shifted version are applied to the redundant observation and effective operating cores, respectively, while the IP core outputs are taken from the operating IP core. Trojan detection circuitry is inserted during the design phase to monitor the redundant observation IP core at run-time. Once a Trojan is detected in the redundant core, the operating IP core is suspended or the identified triggering inputs are isolated for a specific time duration to bypass the Trojan activating trigger. Identified input values and sequences responsible for Trojan activation are stored in order to be filtered out in the future.

In this paper, we present a proof-of-concept application of SECRET to an Advanced Encryption Standard (AES) cryptographic IP core. We introduce our design of an HTH inserted into the IP core at the RTL level and the DFTS monitor to detect the Trojan. The proof-of-concept design is implemented and validated on a Spartan-3 XC3S5000 FPGA, functional feasibility and effectiveness of SECRET are established, and the architecture is characterized in terms of the overheads. The remaining of this paper is organized as follows: a background on HTH DFTS techniques with emphasis on HTH countering methods is presented in Section II. We advance the underlying assumptions, threat model, high-level architecture, and timing analysis of SECRET in Section III. SECRET application to a crypto 3PIP core containing an HTH of our design is advanced in Section IV. SECRET simulation and implementation results for the crypto core are presented in Section V. Conclusions and future work directions are portrayed in Section VI.

II. Background

3PIP cores are widely deployed in reconfigurable systems. Although ideally these cores would be verified by a trusted party in the pre-deployment phase, cost and productivity requirements can make such a development model impractical. Recent progress in DFTS has been made to embed security-enhancing, application-specific, run-time protection circuits within a system to establish tailored trustworthy computing bases. Many run-time Trojan detection methods have been presented in the literature [3]. In this section we discuss some related DFTS efforts with an impact to Trojan countering.

Abramovici and Bradley identified that no existing in the pre-deployment phase mechanisms can guarantee detection of all HTHs and they propose an application-dependent security infrastructure monitoring datapath signals for illegal behaviors. In this approach, reconfigurable Design-for-Enabling-Security (DEFENSE) logic is added to the functional design to implement run-time security monitors [5]. The signals are selected by a designer directly in the RTL and grouped to create multiplexed probe networks sourcing information to security monitors. The hardware-based monitors are configurable finite-state machines that check the current set of signals for behavioral properties specified by the designer. Probe and monitor configurations are controlled by a security control processor, which may also initiate countermeasures implemented by controlling specified datapath signals. When a security violation is detected, the software control processor may override signals or take actions to isolate the core. However, the authors acknowledge that broader countermeasures are required to create a system-level protection scheme. In general, it is an intractable problem to create real-time countermeasures tailored specifically to every possible attack on every system interface. Therefore, the security designer must also be given the flexibility to create abstract countermeasures or real-time enforcement practices that align with system specifications.

Hicks *et al.* propose a hybrid hardware/software Trojan detection and countermeasure defense strategy named BlueChip combining both design- and run-time components [6]. During design-time, Bluechip invokes an Unused Circuit Identification (UCI) algorithm to identify suspicious circuitry. BlueChip provides a detour around suspicious hardware by removing the suspicious circuitry and replacing it with a hardware logic that triggers a software routine implementing the same functionality of the removed hardware. Blue-chip is one of the few methods attempting to provide effective countermeasures against HTH threats. However, software replacement of suspicious hardware circuitry is not the best measure because, usually, the software performance is worse the hardware requirements of the original circuitry. Moreover, despite the complexity of the Bluechip approach, some HTHs can evade detection by the UCI algorithm as stated in [7] and BlueChip might add unnecessary overhead for false positives.

Waksman *et al.* present a solution for disabling HTHs that instead of trying to detect inserted hardware Trojans, they scramble inputs at run-time to prevent the Trojan from receiving its trigger [8]. The authors use three techniques to prevent Trojan activation: applying periodical power resets to prevent triggering time bombs; using data obfuscation to encrypt input values to untrusted units to prevent HTHs from recognizing data-based triggers; and sequence breaking by scrambling the order of the inputs entering the hardware component to prevent HTHs from recognizing their sequence-based triggers. However, this method only provides probabilistic security guarantees about preventing activation of design-level HTHs. Furthermore, this solution solely focuses on input-triggered HTHs leaving the chance for internally-triggered Trojans to easily conduct their tasks.

Kim *et al.* introduce a set of design methodologies and practices to enable operation recovery of hardware components employed in reconfigurable platforms and infected with HTHs by applying Dynamic Partial Reconfiguration (DPR) [9]. They propose replacement of modules infected with HTHs during run-time to enable operation continuity despite the attack. The method is based on a modified system architecture, including an embedded reconfigurable logic, bus architectures for isolating infected hardware modules and maintaining the system performance, and hardware controllers to manage DPR with reliable interface signaling. The authors describe a set of practices of utilizing reconfigurable logic to regenerate system functionality while minimizing the performance degradation and keeping seamless operation under attacks. Unfortunately, the DPR design flow has been immature yet and the associated architectural restrictions and timing considerations can limit the applicability of this method. Furthermore, this method assumes the existence of diverse alternatives for the suspected cores which, if satisfied, can increase the design cost.

III. SECRET Basics and High-Level Architecture

We aim at providing a run-time DFTS countermeasure against HTHs deliberately embedded in 3PIP cores employed in reconfigurable hardware designs. Most DFTS techniques address how to detect HTHs at run-time yet few of them answer the question of how to deal with the detected Trojans to continue the circuit operation securely, or in other words can we design Trojan-tolerant circuits. The answer of the previous question depends on the circuit's ability to continue its operation under attack while reducing or eliminating the Trojan effects. In this paper, we present the SECRET countermeasure in which an HTH can be tolerated in a smart way by allowing the Trojan to be activated at run-time while isolating and suppressing the Trojan payload and going back in time to suspend the identified activating trigger if possible. Prior to describing the SECRET approach to realize such a time travel to the past, the threat model considers the following:

- The threat model considers a 3PIP core containing an HTH and is implemented in a reconfigurable platform.
- The proposed SECRET countermeasure only addresses usually-off rarely-triggered HTHs evading detection by design-time Trojan detection methods. Always-active Trojans with a permanent payload cannot be countered using the proposed solution.
- The threat model does not impose constraints on the Trojan triggering conditions or the payload type. The Trojan might be activated by either a specific input combination, a sequence of inputs or internal state changes, or even after a specific operation time, and its payload might range from functionality change to denial-of-service.
- The protected 3PIP core can be hierarchically instantiated as a top-level module or a lower-level component.

The underlying assumptions of SECRET are listed below:

- We assume that the protected 3PIP core is equipped with DFTS monitors and wrappers that can detect HTHs at run-time. DFTS Trojan detection methods are the key-enablers of SECRET, yet they are not the main interest of this work. The successful operation of SECRET mainly relies on effective detection of HTHs at run-time.
- We assume that the Trojan detection circuitry can detect HTHs inserted at various hierarchical levels and having different triggering conditions and payload types.
- Another assumption is that the Trojan detection circuitry can detect Trojan payloads affecting either the IP core outputs or internal states. Also it might detect either the Trojan trigger or the Trojan payload at run-time. The effectiveness of the SECRET countermeasure is tied to and confined by the detection scheme capabilities.
- We assume the black-box model of the protected 3PIP core indicating that, unlike most Trojan detection solutions, the SECRET countermeasure does not need golden references. However, we should point out that the underlying DFTS Trojan detection approach might require the existence of a golden reference.
- Also we assume that the protected 3PIP core can tolerate output delay and discontinuity for a short time duration.
- Finally we assume that resources overhead translated into silicon area is a cheap price that can be paid to apply sufficient countermeasures enabling operation continuity of 3PIP cores deployed in security-critical applications.

Security-through-diversity solutions rely on multiple diversified instances of a system or component, realizing the same functionality yet developed and outsourced by different parties, accompanied by a majority voting mechanism to detect the odd behavior of the underlying components. Unlike security-through-diversity techniques widely used in the hardware Trojan detection methods [6], [10], [11], our approach employs a redundant or identical instance of the 3PIP core to counter HTHs potentially embedded in the core. This implies that no additional costs are needed to purchase diverse 3PIP cores realizing the same functionality. The challenge becomes how to utilize a redundant instance of the IP core containing the same Trojan infections and running under the same inputs and conditions to overcome and counter the Trojan effects.

Figure 1 shows the high-level architecture of SECRET. Two identical instances of the protected 3PIP core are utilized: an effective operating instance and a redundant observation instance. The IP core input is directly applied to the observation core which is monitored by the DFTS run-time monitors and wrappers having access to the core's inputs, outputs, and internal states. The observation core is redundant and its output is only accessible by the Trojan detection circuitry. The operating core inputs are connected to the time-shifted version of the IP core inputs. A controllable time-shift or delay can be created using a FIFO buffer or a bunch of cascaded registers.

A security controller of our design is the SECRET link to the DFTS circuitry monitoring the observation core in which Trojan detection alarms are raised at run-time. It also acts as an actuator that properly tunes and adjusts SECRET parameters and promptly applies appropriate countermeasures accordingly. The SECRET security controller has control over the time-shift duration between the two cores, passing/isolating the delayed inputs to the operating IP core, and suspending or disabling the operation of the operating core. The time shift between the redundant observation and effective operating IP

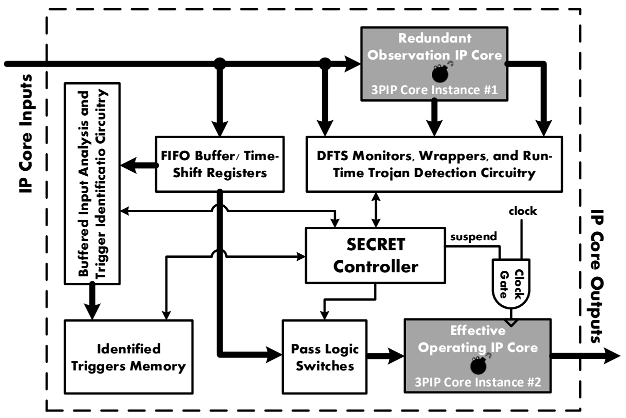
IP Core Inputs
Buffered Input Analysis and Trigger Identificatio Circuitry
FIFO Buffer/ Time-Shift Registers
Redundant Observation IP Core
3PIP Core Instance #1
DFTS Monitors, Wrappers, and Run-Time Trojan Detection Circuitry
SECRET Controller
suspend
clock
Clock Gate
Identified Triggers Memory
Pass Logic Switches
Effective Operating IP Core
3PIP Core Instance #2
IP Core Outputs

Fig. 1. SECRET High-level Architecture

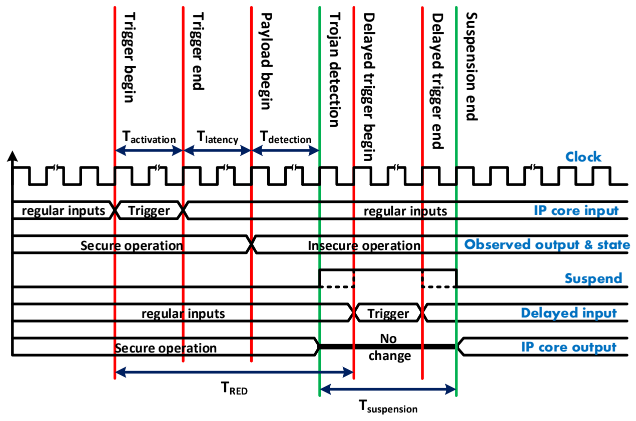

Fig. 2. Timing diagram of SECRET countermeasures

cores T_{RED} (Redundant-Effective Delay) is controlled by adjusting the FIFO length or the number of shift register stages. To control connectivity of the delayed input to the operating core, a set of pass logic switches that can isolate a specific input is inserted under the control of the SECRET controller. A simple AND clock gate can be used to suspend the operating core if needed under the control of the SECRET controller. The overheads incurred by the SECRET architecture include the DFTS security monitors, the redundant IP core, the security controller, and the delay registers and switches.

Figure 2 is a timing diagram illustration of the operation of an IP core containing an HTH and protected with SECRET. This timing diagram provides a typical behavior of a generic usually-off HTH with a rarely occurring trigger equipped with a DFTS Trojan monitors that can detect the Trojan payload after a specific time of its start. Vertical lines in red indicate events caused by the Trojan with unknown occurrence time. The activation time $T_{activation}$ is the time needed to activate the Trojan either externally via the core inputs or internally via state changes, where $T_{activation}$ equals zero for data-triggered Trojans and is greater than zero for sequence-triggered and time-activated Trojans. A latency time $T_{latency}$ is considered between the triggering sequence end and the payload beginning, which equals to zero if the payload begins immediately after the trigger ends. We assume that the DFTS monitor needs a duration of $T_{detection}$ from the payload beginning to detect the Trojan. In this scenario, we do not consider the case in which the DFTS monitors can detect the Trojan trigger rather the payload, i.e. we only considered the worst case scenario where the Trojan will be detected after its payload begins.

Inputs to the operating IP core are delayed by a time duration of T_{RED}. The basic underlying idea of SECRET is that if we can isolate identified input triggers to the operating core or suspending its operation, if the Trojan is triggered internally or if the input trigger cannot be identified, during the triggering sequence, the Trojan activation condition will be bypassed and the IP core can continue its secure operation. The suspension time $T_{suspension}$ can immediately start after Trojan detection. T_{RED} and $T_{suspension}$ are the two main parameters of the SECRET countermeasure. As demonstrated by Figure 2, two conditions must be satisfied to circumvent the Trigger:

1) The Trojan must be detected before the delayed trigger to the operating core begins. This condition can be mathematically formulated as:

$$T_{RED} \geq T_{activation} + T_{latency} + T_{detection}$$

2) Assuming the first condition is satisfied, the monitoring IP core must be suspended during the delayed activation duration to completely avoid the triggering sequence. This condition can be mathematically formulated as:

$$\begin{aligned} T_{suspension} &\geq T_{RED} + T_{activation} \\ &\quad - (T_{activation} + T_{latency} + T_{detection}) \\ &= T_{RED} - (T_{latency} + T_{detection}) \end{aligned}$$

T_{RED} and $T_{suspension}$ are the two SECRET parameters that can be set to fixed values in the design-time to circumvent the Trojan trigger in the effective operating core. Unfortunately, SECRET parameters are functions of the unknown Trojan and DFTS detection circuitry timing parameters. Compared to time travel to the past, T_{RED} is the time duration in which we can go back in time to prevent triggering detected HTHs in the present. In the given scenario, unfortunately, we do not know precisely the time of the event —the onset of the trigger or payload— we aim to avoid. If the DFTS monitors can detect the triggering onset, we can precisely set both SECRET parameters. Information gained from the design-time analysis of the IP core associated with the core application context and the DFTS monitor timing characteristics can provide upper-bound estimates of the required timing parameters.

IV. SECRET Application to a 3PIP Crypto Core

As a proof of concept, we apply SECRET to an open source AES cryptographic 3PIP core that works on a fixed data size of 128-bit and uses a key size of 128, 192 or 256 bits [12]. Figure 3 shows a block diagram of the target Crypto core. The key length is determined at compile-time and is kept fixed at run-time. The bus cycle is 10, 12, or 14 clock cycles based on the key size, which determines the number of encryption/decryption rounds. The IP core works as a memory-mapped processor peripheral with address space of 32 words each of 32-bit width. This memory is accessible through an address port, and the address space is divided into 4 parts:

- The first part is a write-only key space that is 4, 6 or 8 words based on the key size selected at compile-time.
- The second part is the input data space that consists of 4 write-only words.
- The third part is a 4-word read-only result space.
- The last part is a single read/write control word that contains only 4 bits and the rest is reserved.
- The remaining memory words are reserved.

The core operates in the following sequence: First the user key and input data are written to the key space and data space in the memory, respectively. Before a `key_valid` bit in the control word is set to start key expansion, and a `dec` or `enc` bit is set to start the encryption/decryption operation. Both key expansion and encryption can start simultaneously by setting the `key_valid` bit, which has to be maintained high during the operation. A counter is used as an address to select the round key. The total number of processing rounds is a constant defined by the key size. The result is stored in the result space in the memory where it can be read through the read data port.

A. *Trojan Implementation*

HTHs must be implemented such that it cannot be activated during all validation and testing phases. This requires the Trojan to have the smallest effect possible on both area and power consumption, and also requires that its trigger is very difficult to detect, implying that it should depend on more than one unlikely condition. We note that the read and write selects are mutually exclusive and should not be asserted together. When reading the write-only memory space the control word is read instead, and the reserved memory areas are read as zeros. Taking advantage of these two notices we implemented our HTH at the RTL level to satisfy the previous conditions.

We set the trigger to be a combination of two unlikely conditions. The first is a string of zeros and ones written to the reserved part of the control word, this reserved part has no effect on the normal core operation, and thus cannot be detected at run-time, and the random selection of this string makes it extremely unlikely to detect. The second condition is to simultaneously assert both read and write selects. When both conditions are satisfied, the write-only key space is converted to a read/write space for n-bus cycles and the key can then be leaked during this period. To minimize the Trojan size and make it difficult to detect, the key space will be converted to a read/write space for only a single bus cycle. Figure 3 shows the HTH triggers and payload in the AES core. The Trojan has only a 1.4% and 0.17% resource and power overhead, respectively, as shown in Table I .

B. *Run-time Detection and SECRET Logic*

The most important assumption for successful operation of SECRET is to detect potential HTHs at run-time. Despite Trojan detection is not our main goal, we design a run-time monitor that can detect the implemented HTH. Detection of the implemented HTH is done by monitoring all transactions on inputs and outputs, and detecting illegal transactions. For the selected crypto core, an example of illegal transactions

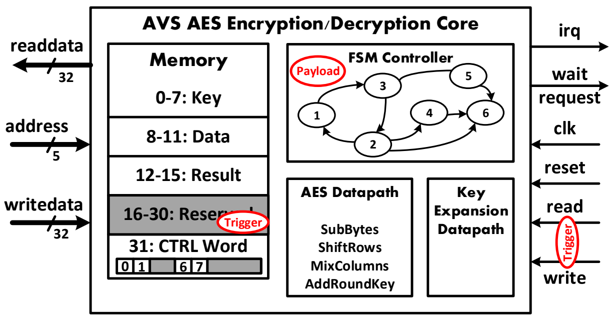

Fig. 3. Block Diagram of the target encryption/decryption core

would be a read request to a write-only memory space that returns something other than the control word, or a read transaction to the reserved memory part that returns a non-zero value. The Trojan detection circuitry is implemented as hardware assertion checkers [13].

Our detection logic comprises a small memory space that holds a copy of the key, assertions that compare output data with individual words of the stored key and assertions that compare the output data to the control word format. This detection circuit is very simple and effective for detecting specific HTHs, namely, those who would leak the encryption key. The aforementioned HTH and the simple detection circuitry are sufficient for the SECRET demonstration purpose.

In the SECRET architecture, we set T_{RED} to four bus cycles which introduces a four-bus-cycle delay to read transactions instead of zero bus cycles delay. This delay is a SECRET parameter and it should not pose a problem for streaming applications using the crypto IP core. Once a Trojan is detected, the recovery process immediately begins. The operating core is disabled for a specific time $T_{suspension}$, preventing the operating core from receiving the attack trigger. $T_{suspension}$ is another SECRET parameter selected to ensure that the trigger is prevented and no unnecessary transactions are stopped. In this specific application, the operating core is disabled for two bus cycles which is an enough time to prevent at least one of the two conditions required to activate the Trojan. After the disable signal is deasserted the delayed core continues its operation normally. During this time a `Security_Emulate` signal is asserted to flag that the core is suspended.

At the same time, a trigger analysis module checks the input FIFO buffer attempting to identify the trigger. This process runs concurrently to the recovery process and it can prevent future attacks exploiting the same Trojan. Deeper buffers can be used to increase the probability of detecting the trigger at the expense of increased output delay and resource overheads. If any abnormal input values or sequences are detected, they are stored in a dedicated memory, and in the future all inputs are compared to the memory contents and isolated if matched.

V. Results and Evaluations

Figure 4 shows the simulation results of the SECRET application to the crypto 3PIP core. It illustrates the trigger beginning and end, where the trigger control word is written and the read and write signal are applied together. The payload begins after one cycle of the Trojan trigger sequence. The Trojan is detected in the same bus cycle where the Trojan payload begins. The buffer delay T_{RED} is four bus cycles, and the suspension time $T_{suspension}$ is two bus cycles.

The AES crypto core is synthesized using Xilinx tools on a Spartan-3 XC3S5000 FPGA and at a clock frequency of 100 MHz. Our Trojan is implemented at the RTL level by modifying the HDL source code describing the IP core. Table I depicts the synthesis results of the 3PIP core, HTH, detection and SECRET logic. The results depict an increase of only 3.12% in the number of LUTs and a 0.43% increase in power consumption for the infected core compared to the Trojan-free implementation. These results demonstrates the difficulty of detecting the implemented HTH using design-time techniques.

Adding the detection and SECRET protection logic containing the redundant IP core and SECRET logic leads to a significant increase in the resources, as expected. The added detection and recovery logic are responsible for increasing power consumption by 22.45%. The selected length of the delay buffer has a direct impact on resources and power consumption. The redundant core which is always running and, consequently, consumes more power than the operating one which is suspended under attacks. The resource and power overheads are highly dependent on the IP core itself, the employed detection method, and the SECRET parameters.

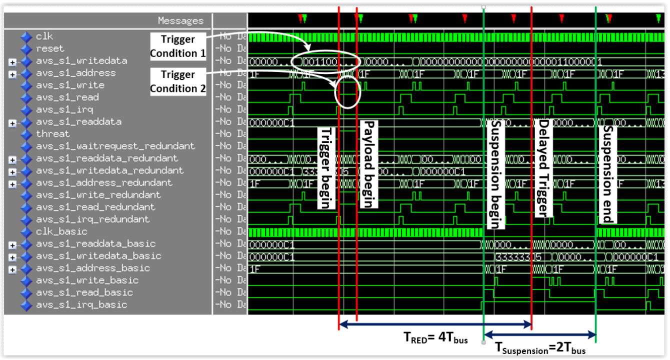

Fig. 4. Simulation of SECRET application to the AES Crypto 3PIP core

TABLE I
SYNTHESIS RESULTS OF THE IP CORE, HTH, AND SECRET LOGIC

Design	Registers		LUTs		Power	
	#	%	#	%	mW	%
Trojan	1	0.017	99	1.43	0.92	0.1764
Operating core	2761	46.3	3169	45.5	193.72	37.15
Redundant core	2761	46.3	3169	45.5	209.61	40.2
SECRET Logic	440	7.383	527	7.57	117.09	22.45
Total	5962	100	6964	100	521.34	100

VI. CONCLUSIONS

In this work we advanced a novel approach that exploits circuit redundancy to counter hardware Trojan threats in 3PIP cores deployed in reconfigurable hardware designs. SECRET employs two identical instances of the protected IP core, one for observation and the second with delayed input for operating. Once a Trojan is detected in the redundant observed core, the operating core is suspended or the input triggers are isolated. We presented the SECRET architecture, timing behavior, and main parameters. A proof-of-concept application to a 3PIP crypto core with an embedded HTH was introduced and the simulation and validation results establish the SECRET feasibility and effectiveness. The implementation results depict the increased overhead incurred by using a redundant instance of the protected IP core. The SECRET protection is highly dependent on the target IP core and the Trojan design yet the architecture and DFTS practices presented herein can be generally applied to protect a broad range of 3PIP cores widely deployed in reconfigurable hardware designs.

This work provides an initial effort to evaluate the proposed SECRET countermeasure which can be merged with related Trojan detection methods to produce a very efficient Trojan detection and countering solution. Many research points can emanate from this work including precise characterization of the SECRET parameters, identifying qualified run-time Trojan detection techniques to enable SECRET, confining potential target IP cores and HTH threats, studying run-time trigger identification algorithms, and automation of the SECRET logic design process. In our future work, we will also investigate reducing overheads incurred by using a redundant instance of the protected IP core. We will also investigate dynamic tuning of the SECRET parameters to increase the probability of countering various types of HTHs.

REFERENCES

[1] John Koeter. Tomorrow's semiconductor IP - not business as usual. *Chip Design Magazine*, 2013.

[2] M Tehranipoor and F Koushanfar. A survey of hardware trojan taxonomy and detection. *Design Test, IEEE*, PP(99):1–1, 2013.

[3] Mark Beaumont, Bradley Hopkins, and Tristan Newby. Hardware trojans-prevention, detection, countermeasures (a literature review). Technical report, DTIC Document, 2011.

[4] Mohammed M Farag. *Architectural Enhancements to Increase Trust in Cyber-Physical Systems Containing Untrusted Software and Hardware*. PhD thesis, Virginia Polytechnic Institute and State University, 2012.

[5] Miron Abramovici and Paul Bradley. Integrated circuit security: new threats and solutions. In *Proceedings of the 5th Annual Workshop on Cyber Security and Information Intelligence Research: Cyber Security and Information Intelligence Challenges and Strategies*. ACM, 2009.

[6] Matthew Hicks, Murph Finnicum, Samuel T King, Milo Martin, and Jonathan M Smith. Overcoming an untrusted computing base: Detecting and removing malicious hardware automatically. In *Security and Privacy (SP), 2010 IEEE Symposium on*, pages 159–172. IEEE, 2010.

[7] Cynthia Sturton, Matthew Hicks, David Wagner, and Samuel T King. Defeating UCI: Building stealthy and malicious hardware. In *Security and Privacy (SP), 2011 IEEE Symposium on*, pages 64–77. IEEE, 2011.

[8] Adam Waksman and Simha Sethumadhavan. Silencing hardware backdoors. In *Security and Privacy (SP), 2011 IEEE Symposium on*, pages 49–63. IEEE, 2011.

[9] Lok-Won Kim and J.D. Villasenor. Dynamic function replacement for system-on-chip security in the presence of hardware-based attacks. *Reliability, IEEE Transactions on*, 63(2):661–675, June 2014.

[10] D McIntyre, F Wolff, C Papachristou, Swarup Bhunia, and D Weyer. Dynamic evaluation of hardware trust. In *Hardware-Oriented Security and Trust, 2009. HOST'09. IEEE International Workshop on*, pages 108–111. IEEE, 2009.

[11] Hany AM Amin, Yousra Alkabani, and Gamal MI Selim. System-level protection and hardware trojan detection using weighted voting. *Journal of Advanced Research*, 5(4):499–505, 2014.

[12] Thomas Ruschival. Avalon AES ECB-Core (128, 192, 256 bit). http://opencores.org/project,avs_aes, 2009.

[13] Marc Boulé and Zeljko Zilic. *Generating hardware assertion checkers*. Springer, 2008.