Hardware Trojan Detection using ATPG and Model Checking

Jonathan Cruz

Department of Computer and Information Science and Engineering
University of Florida, Gainesville FL 32611-6120, USA

Bachelor of Science in Computer Engineering, Spring 2017

Summa Cum Laude

Advisor: Prof. Prabhat Mishra

***Abstract*— This honors thesis deals with the analysis of using automatic test pattern generation (ATPG) and model checking approaches for hardware Trojan detection. Specifically, I investigate their effectiveness in full-scan and partial-scan designs. A novel approach based on combining these two techniques is introduced in dealing with designs in which ATPG and model checking are expected to fail when applied alone. The approach works best in designs with partial-scan chain insertion. Using model checking on non-scan and ATPG on full-scan partitions avoids common pitfalls of running the full design using any one of these techniques. The resulting test vectors generated by combining these approaches can then be used to detect Trojans.**

I. Introduction

Designing today's system-on-chips (SoC) is a highly complex process that involves many time-to-market constraints. It is common practice to integrate third-party IPs in the production of SoCs in order to remain competitive in today's global market. However, interfacing with third-party IP raises concerns, as now an integral part of the SoC's design process is no longer under complete scrutiny. An adversary within these third-party facilities can tamper with the design or insert malicious IP, also known as a Trojan. Hardware Trojan attacks range from information leakage to complete chip malfunction. Detection of Trojans inserted at the pre-silicon phase of the fabrication process is extremely difficult and often missed during testing. Furthermore, with the introduction of partial-scan designs, it becomes even harder to detect Trojans with common verification techniques alone [1] [2] [3].

This paper investigates the use of ATPG and model checking tools on soft IP in an effort to detect hardware Trojans in scan-chain inserted designs. A novel approach that combines the strengths of two common design verification techniques, already a part of the normal verification design flow, is introduced to increase the detection rate of Trojans in partial-scan designs. The rest of this honors thesis is presented in the following manner: Section II describes the background and related work. Section III introduces the combined approach of using ATPG and model checking. Section IV describes the experimental setup, analysis, and results. Finally, Section V concludes the thesis and suggests future work.

II. Background and Related Work

A. Design For Testability

Design for Testability (DFT) is a technique employed in designing ICs for the purposes of reducing test costs and time [1]. A very common technique for DFT is scan-chain insertion. The idea here is to replace flip-flops (FFs) in the design with scan flip-flops – a flip-flop with a multiplexer attached to the input. The two inputs to this multiplexer are the original input and a new scan-in input. A select line, scan enable (SE), chooses between these two inputs; when SE is high, scan-in values are selected, else the original input is sent to the FF. Scan FFs can be chained together as shown in Figure 1 to form a scan-chain. The purpose of these scan-chains is to reduce the loading time when testing sequential elements of a design, increasing a circuits controllability and observability.

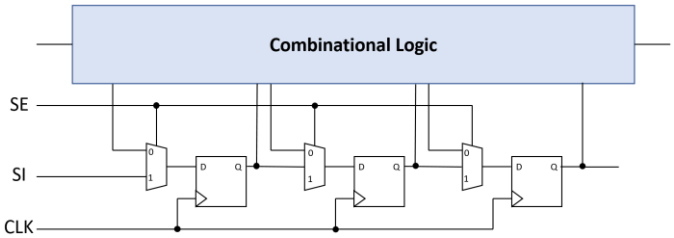

Figure 1

While including a complete scan-chain is ideal, it may not be possible due to the incurred area overhead or delays that invalidate various design constraints. Partial-scan chain insertion is used as a cost-effective alternative [4] to achieve acceptable test coverage, while maintaining design constraints. In partial-scan designs, not all FFs are included in the scan-chain. However, these non-scan FFs introduce branches in the circuit with low controllability and observability, which can be exploited as a rare trigger condition by a malicious attacker.

B. Automatic Test Pattern Generation (ATPG)

ATPG is a verification methodology used to generate test vectors that allows designers to identify faulty design behavior in a timely manner [1]. The goal of ATPG is to create a set of test patterns that achieve a desired test coverage, TC, and fault coverage, FC, through fault simulation:

$$TC = \frac{Number\ of\ detected\ faults}{Number\ of\ testable\ faults}$$

$$FC = \frac{Number\ of\ detected\ faults}{Number\ of\ total\ faults}$$

With the use of DFT, ATPG can efficiently generate test vectors by treating the any design as combinational logic. This is no longer the case in partial-scan designs. ATPG tools must now consider a set of test vectors that will activate a target fault. The worst-case complexity (cyclic sequential designs) of sequential ATPG becomes $9^{N_{ff}}$, where N_{ff} is the number of flip-flops [5].

1) TetraMAX

Synopsys' TetraMAX is a verification tool used for automatic test pattern generation [6]. Figure 2 describes the tool flow for TetraMAX ATPG. TetraMAX reads in a gate-level netlist along with its corresponding libraries which are used to build the design locally. A Standard Test Interface Language (STIL) file generated by the DFT compiler is read and used in the design rules checking phase (DRC). The DRC process verifies that the physical layout of the design meets fabrication constraints. After the DRC phase is passed, ATPG can begin. A fault list, either produced internally or provided through an external file, is specified by the user in preparation for ATPG.

This list contains information on the types of fault to be tested [1] and their locations in the design. ATPG can then be performed in one of three modes: basic-scan, fast-sequential,

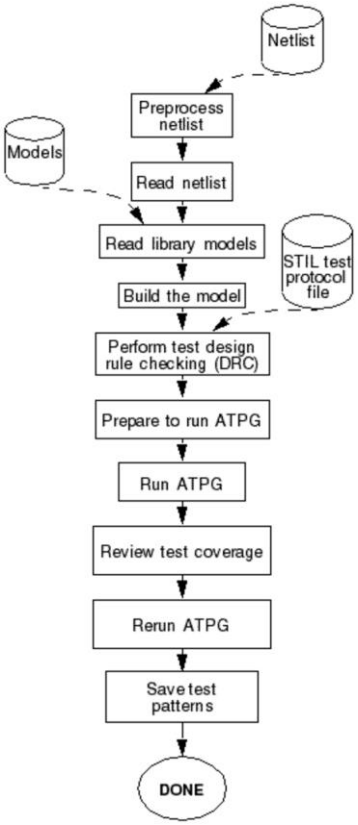

Figure 2

and full-sequential. In basic scan, TetraMAX operates as full-scan combinational ATPG tool resulting in high test coverage. Designs run in basic scan mode must be full-scan. Both fast and full sequential ATPG modes provide support for partial-scan designs by executing capture procedures in between scan chain load and unload phases. The difference between fast and full sequential ATPG modes is that all clock and reset signals to non-scan elements must be controlled at the primary input in fast-sequential ATPG.

As with any ATPG tool, sequential test pattern generation is a complex process [5] [7]. While scan-chains are implemented to mitigate the complexity, designs with a significant amount of non-scan cells can greatly reduce ATPG performance, resulting in ATPG-untestable faults [9] that an attacker can exploit.

C. Model Checking

Model checking is a formal method used to verify a design against functional properties (expected design behavior). Given a design and a list of properties expressed in temporal logic, a model checker will either verify the design property or generate a counter-example that shows why the property isn't satisfied [18]. This approach can be extended for use in verifying design security [19]. Properties can be written and verified with security features in mind, such as monitoring the primary output for leaking secret keys [10]. However, a common problem of the model checking approach is state explosion caused by the exponential nature of exploring a design's state space. This fact limits the practicality of model checking on larger designs.

1) Symoblic Model Verifier

Symbolic Model Verifier (SMV) is a model checking tool developed by Ken McMillan for verifying FSMs using Linear Temporal Logic (LTL) or Computation Tree Logic (CTL) [11]. SMV's algorithm uses ordered binary decision diagram (OBDD) and bounded model checking (BMC) in an effort to reduce state explosion while exploring the state space of the design.

To verify a design, users must first either manually, or through the use of a program, translate their design into a model specification language understood by the tool. Design properties are then described in CTL or LTL. Once the tool has both the design and properties to be verified, it begins to unroll the state space. A Boolean satisfiability assignment is extracted from the unrolled states and checked using an internal SAT solver. If unsatisfiable, the model checker produces a counterexample computation path.

D. Trojan Detection

Both Wolff et al. and Zhang et al. propose Trojan detection approaches that incorporate ATPG tools for generating test patterns [12] [20]. Yet, with the introduction of partial-scan designs the effectiveness of full-sequential ATPG for generating test patterns is greatly reduced due to the complexity of full-sequential ATPG on non-scan FFs.

The method proposed in [8] utilizes N-detect full scan ATPG and SAT solver for Trojans detection. However, this approach also fails to effectively consider designs that have a significant non-scan portion which will limit the effectiveness of ATPG.

III. Trojan Detection Using ATPG and Model Checking

I explore the suitability of using ATPG and model checking for hardware Trojan detection. Two types of designs are considered in my analysis: full-scan and partial-scan designs. In both situations, analysis must be performed at the RTL level to determine suspicious gates in the design. Suspicious gates are identified through rare branch coverage. More specifically, in a design, rare branches are branches that are not covered after running random tests up to millions of cycles. These branches identified as 'rare' will be used in model checking property generation and ATPG stuck-at faults. The rationale here is that a Trojan will be activated as a result of a rare sequence of inputs and/or state transitions; otherwise, the malicious insertion becomes a triviality that can be detected during design testing and verification [12]. Other statistical methods for determining rare branches are also suitable such as FANCI [13] and MERS [14].

A. Full-Scan Designs

In designs with a full-scan chain, all sequential gates not assumed any sequential logic that is inserted as part of the Trojan circuit is non-scan in order to reduce its detectability

through conventional test methods. For ATPG analysis, test vectors are generated from stuck-at faults that are tested at all gates in the design identified from the set of rare branches. Yet even 100% fault coverage cannot guarantee complete detection of hidden Trojans [1].

With model checking, properties for triggering the suspicious branches are expressed in LTL and run through the model checking tool. The negation of the rare branch trigger is defined as a property, therefore the model checking tool will generate a counter-example to trigger each branch.

B. *Partial-Scan Designs*

As previously discussed, designs in which not all sequential elements are included in the scan-chain are considered partial-scan designs. The non-scan FFs can be ideal candidates for embedding Trojans due to the low controllability and observability.

Similar to full-scan designs, the partial-scan designs are run through ATPG and model checking tools to generate test vectors for activating rare branches. ATPG is expected to fail in most partial-scan designs with significant sequential depth, as previously detectable faults can be unintentionally rendered undetectable with the removal of scan FFs. Model checking still suffers from state explosion.

Combining ATPG and model checking shows promising results. Figure 3 describes the method for combining the two techniques. First, a design is compiled with partial-scan insertion. The design is then divided into its full-scan and non-scan partitions. But before the non-scan partition is run through a model checking tool, security properties must be written using temporal logic. These security properties are simply trigger conditions for rare branches identified in the design from the rare branch marking phase. Properties are then written to activate the rare branch using LTL statements [18].

The model checking tool will then generate a set of counter-examples that activate the rare branches from the input of the non-scan partition. Each counter-example is translated into a combinational stuck-at fault circuit inserted into the full-scan design, creating what I refer to as a full-scan* design. This at most adds an overhead of N+3 gates [17], where N is the number of primary outputs from the non-scan design. Afterwards, the full-scan* design is run using an ATPG tool to generate test vectors that trigger the stuck-at faults.

The main advantage of using this approach is circumventing the downsides of ATPG and model checking. With only the full-scan partition, the ATPG tool experiences a much faster execution time as the complexity of non-scan full-sequential ATPG is removed. Likewise, because the model checking tool is only given a subset of the design, the state space explored is smaller mitigating the problem of state explosion. However, the state space in the non-scan portions can still be significant. Dividing the non-scan partitions even further can help in such cases.

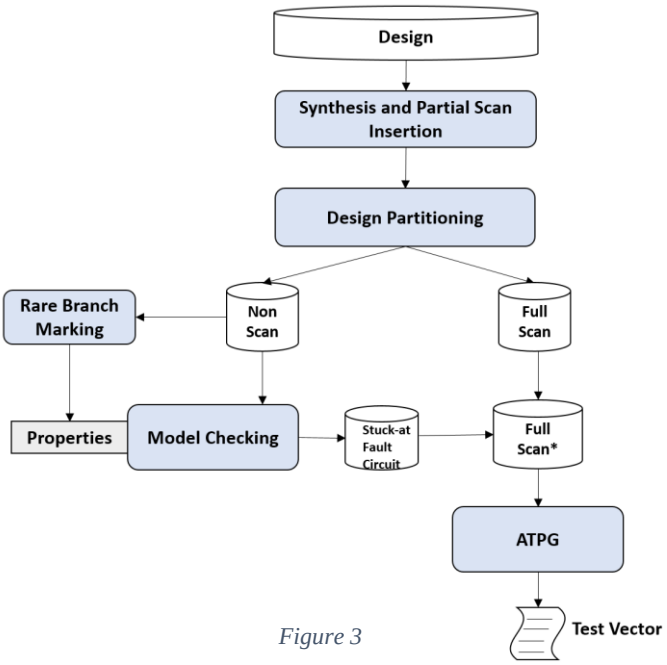

Figure 3

C. *Golden Model*

The test vectors resulting from the ATPG tool, model checker, or the combined approach are translated into test benches. Both the golden and suspicious designs are run under the testbench and their outputs compared (XOR) to detect the presence of a functional Trojan.

IV. Experiments

A. *Experimental setup*

In my investigation, only rare-event triggered Trojans are considered. Therefore, AES-128 and RS232 benchmarks from Trust-Hub benchmark suite are used. More information on the Trojan circuits and their implementation can be found on Trust-Hub [15] [16]. Additionally, two modified AES benchmarks (cb_aes) are used to showcase the limitations of model checking and ATPG.

A machine with Intel Core i5-3470 CPU @ 3.20GHz and 8 GB of RAM is used for testing. The benchmarks are synthesized using design compiler with DFT scan insertion. For full-scan designs, the original RTL design is synthesized with full-scan insertion. With partial-scan designs, a subset of the FFs is selected for scan in such a way that maintains a high test coverage. The purpose of these scan insertion techniques is to simulate a scenario in which an adversary would insert a Trojan in hard to detect areas after scan-chain insertion and during integration with third party IP. The SMV tool [11] is used for model checking and Synopsys TetraMAX [6] for ATPG. Some constraints were imposed on the design due to tool limitations. For example, in SMV results are not accurate

Full-Scan Benchmarks	Test Coverage	# Rare branches	ATPG		Model Checking	
			Detected	CPU Time	Detected	CPU Time
AES-T1000	99%	2	X*	0.02s	X*	140.8s
AES-T2000	99%	5	X*	0.95s	X*	333.35s
RS232-T400	99%	2	√	0.02s	√	1 hour
RS232-T800	99%	1	√	0.02s	√	636.533s
cb_aes_15	99%	1	√	0.27s	X	BDD limit
cb_aes_20	99%	1	√	0.27s	X	BDD limit

*activated not propagated

Table 1: Comparison of ATPG and model checking techniques for Trojan detection in full-scan designs

Partial-Scan Benchmarks	% FFs in scan-chain	Test Coverage	# Rare branches	ATPG		Model Checking		Combined	
				Detected	CPU Time	Detected	CPU Time	Detected	CPU Time
AES-T1000	93%	99%	2	X*	0.02s	X*	85.86s	X*	8.8s
AES-T2000	91%	99%	5	X*	0.90s	X*	216.5s	X*	22s
RS232-T400	51%	97%	2	√	0.24s	√	1 hour	√	0.52s
RS232-T800	45%	97%	1	√	0.06s	√	7.233s	√	0.12s
cb_aes_15	85%	99%	1	√	8 hours	X	BDD limit	√	7.85s
cb_aes_20	93%	99%	1	√	8 hours	X	BDD limit	√	38.3s

*activated not propagated

Table 2: Comparison of our approach (combined) with ATPG and model checking for Trojan detection in partial-scan designs

in designs with multiple clock domains and in TetraMAX sequential elements with multiplexed clocking are not allowed.

B. Results

My experimental results from full-scan designs are shown in Table 1. Column 2 reports the calculated test coverage from the scan design. Column 3 shows the number of rare branches identified from the rare branch calculation on the RTL. In columns 5 and 7, the CPU time in generating test vectors for each approach is reported. Columns 4 and 6 show whether the test vectors detected the Trojan in the testbench. As expected, ATPG performs well in designs with full-scan insertion and model checking experiences a BDD Out of Memory limit when running the custom benchmarks due to the significant state space unrolled.

Results from partial-scan designs are described in Table 2. Columns 2 and 3 describe the percent of FFs that are included in the scan-chain and the corresponding test coverage. Table 2 shows the same metrics as Table 1, but now also includes data for the combined approach. The CPU times for model checking in column 8 are less than the times from full-scan (Table 1, Column 7), likely due to increased area and input from the scan-chains. Yet, despite the amount of partial-scan, ATPG still finds a test vector for the AES benchmarks due to the nature of the Trojan, which is controlled by the primary input. The custom benchmarks with partial-scan exploit the weaknesses in ATPG and model checking through a significant non-scan sequential depth. The combined approach's results provide comparable times in most cases or better times in the case of the custom benchmarks where significant sequential depth and width are introduced.

For AES benchmarks, while the Trojan is activated by the test vector, it is not detected by the testbench. The AES Trojan introduces a new output that leaks the key over several cycles, which isn't directly comparable with the golden model.

V. Conclusion

In this honors thesis, I investigated the suitability of using ATPG and model checking for hardware Trojan detection in full-scan and partial-scan designs. Experimental results demonstrated the merits and weaknesses in both approaches and the effectiveness of combining them in partial-scan designs. Future work will include investigating more partial-scan benchmarks with significant sequential depth to further explore the effectiveness of the proposed approach.

Acknowledgment

This research work is supported by NSF REU Supplement #1642267. Without the mentorship and guidance of Dr. Mishra and his PhD students, Farimah Farahmandi and Alif Ahmed, this research would not be possible.

References

[1] M. Tehranipoor and C. Wang, "Introduction to Hardware Security and Trust", Springer, August 2011.

[2] S. Bhunia, M. Hsiao, M. Banga, and S. Narasimhan, "Hardware Trojan attacks: threat analysis and countermeasures", Proceedings of IEEE, vol. 102, pp. 1229-1247, 2014.

[3] M. Tehranipoor and F. Koushanfar, "A survey of hardware Trojan taxonomy and detection", IEEE Design and Test of Computers, vol, 27, pp. 10-25, 2010.

[4] V. Chickermane, J. H. Patel, “An Optimaztion Based Approach to the Partial Scan Design Problem”, Proceedings. International Test Conference 1990, Washington, DC, 1990, pp. 377-386.

[5] M. Bushnell , Vishwani Agrawal, Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits, Springer, 2013

[6] Synopsys Inc. TetraMAX ATPG User Guide Version H-2013.03-SP4, September 2013

[7] T. E. Marchok, A. El-Maleh, W. Maley, J. Rajski, “Complexity of Sequential ATPG” Proceedings of European Design and Test Conference, 1995.

[8] M. Banga and M. Hsiao, ‘‘Trusted RTL: Trojan detection methodology in pre-silicon designs,’’ in Proc. IEEE Int. Workshop Hardware-Oriented Trust Security, 2010, pp. 56–59.

[9] I. Pomeranz and S. M. Reddy, "On Undetectable Faults in Partial Scan Circuits", in Proc. ICCAD-02, pp. 82-86.

[10] J. Rajendran, A. Dhandayuthapany, V. Vedula, and R. Karri, "Formal security verification of third party intellectual property cores for information leakage", International Conference on VLSI Design, pp. 547-552, 2016.

[11] K.L. McMillan. Symbolic Model Checking. Kluwer Academic Publishers, 1993.

[12] F. Wolff, C. Papachristou, S. Bhunia, R. S. Chakraborty, “Towards Trojan-Free Trusted ICs: Problem Analysis and Detection Scheme” Proceedings of the conference on Design, Automation and Test

[13] A. Waksman, M. Suozzo, and S. Sethumadhavan, "FANCI: identification of stealthy malicious logic using boolean functional analysis", ACM Conference on Computer and Communications Security, pp. 697-708, 2013.

[14] Y. Huang, S. Bhunia, P. Mishra, "MERS: statistical test generation for side-channel analysis based Trojan detection", ACM Conference on Computer and Communications Security, pp. 130-141, 2016.

[15] H. Salmani, M. Tehranipoor, and R. Karri, "On design vulnerability analysis and trust benchmark development", IEEE Int. Conference on Computer Design, 2013.

[16] M. Tehranipoor, D. Forte, R. Karri, F. Koushanfar and M. Potkonjak, "Trust-Hub benchmark suite", Available: https://www.trust-hub.org.

[17] Y. C. Kim, V. D. Agrawal, K.K. Saluja, “Multiple Faults: Modeling, Simulation and Test”, 15th Int. Conf. on VLSI Design, 2002, pp. 1-6

[18] E. M. Clarke, O. Grumberg, D. A. Peled, Model Checking, MIT Press, 1999

[19] P. Mishra, S. Bhunia, M. Tehranipoor, Hardware IP Security and Trust, Springer, 2017

[20] X. Zhang, M. Tehranipoor, “Case Study: Detecting Hardware Trojans in Third-Party Digital IP Cores”, Hardware-Oriented Security and Trust (HOST), 2011 IEEE International Symposium, June 2011, pp. 67-70.