Towards Trojan-Free Trusted ICs: Problem Analysis and Detection Scheme

Francis Wolff, Chris Papachristou, Swarup Bhunia, Rajat S. Chakraborty
Case Western Reserve University
Cleveland, Ohio 44106, USA
{fxw12,cap2,skb21,rsc22}@case.edu

Abstract

There have been serious concerns recently about the security of microchips from hardware trojan horse insertion during manufacturing. This issue has been raised recently due to outsourcing of the chip manufacturing processes to reduce cost. This is an important consideration especially in critical applications such as avionics, communications, military, industrial and so on. A trojan is inserted into a main circuit at manufacturing and is mostly inactive unless it is triggered by a rare value or time event; then it produces a payload error in the circuit, potentially catastrophic. Because of its nature, a trojan may not be easily detected by functional or ATPG testing. The problem of trojan detection has been addressed only recently in very few works. Our work analyzes and formulates the trojan detection problem based on a frequency analysis under rare trigger values and provides procedures to generate input trigger vectors and trojan test vectors to detect trojan effects. We also provide experimental results.

1. Introduction

The trust issue is an emerging problem in semiconductor integrated circuit (IC) security. This issue has been raised recently due to outsourcing of the chip manufacturing processes to reduce cost. This may create a major risk for semiconductor systems in critical applications such as communications, space, military and nuclear facilities. The problem appears from the loss of control that traditional semiconductor industry used to have of the manufacturing cycle. It is becoming common practice for the design cycle to be done far away from manufacturing and electronically transferring GDS II files to outsourced fabrication facilities. In one such attack, a trusted design can be tampered in an untrusted fabrication facility with the insertion of malicious circuitry that triggers a malfunction conditionally. This malicious circuitry, referred to as a *hardware trojan*, can trigger and affect normal circuit operation, potentially with catastrophic outcome.

For a trojan circuit to be effective, from the attacker's viewpoint, a) it should be triggered under rare events or conditions, b) it should not be easily detected by regular testing, both functional and ATPG. The trojan detection problem is different from test problem because the trojan effects can not be modeled directly as traditional digital circuit faults. Apparently, there will be a need for trojan detection models, methods and techniques to alleviate this emerging trust issue of semiconductor ICs especially in critical applications. This need has provided the main motivation for this work.

There are two aspects in the trojan mechanism, the input trigger and the output payload. Normally, a trojan is invisible to the circuit operation, meaning triggering is inactive with no payload effect. Under certain conditions or events, the trojan is activated (triggered) and then the payload injects an error that may be latched at the circuit output. There are three categories of trojan triggering: 1) rare value triggered; 2) time-triggered and 3) both time and value triggered. The focus of our work is on rare value type of triggering trojans. Our work is two fold. First, we analyze and investigate the trojan effect. Second, we formulate the trojan detection problem based on a frequency analysis of rare values and provides procedures to generate input trigger vectors and trojan test vectors to detect trojan effects.

The rest of the paper is organized as follows. In Section 2 we discuss related works. In Section 3 we discuss the classification of the trojan circuits and provide examples of different trojan types. A trojan vector generation scheme to detect combinational trojans is presented in Section 4 along with trojan coverage results for combinational benchmarks in Section 5. Section 6 concludes the paper.

2. Background

The problem of TRUST in ICs is fundamentally different from the side-channel attack or scan-chain based attack approaches [2, 4, 5] where the hacker tries to externally decipher a secret key embedded internally in the secured chip, by analyzing a measured physical parameter, the so called "side-channels" such as power, temperature and electromagnetic profiles. In the case of TRUST in ICs, the hacker has internal access to the IC manufacturing facilities and tries to disrupt or compromise the normal functionality of the IC in the field. Furthermore, the hacker will only modify a small random sample of chips in the production line, making it difficult to apply reverse engineering techniques [3] as a general solution. Also, by using a small sample, it becomes difficult to distinguish between chip failure in the field or trojan insertion.

In [1], the authors apply the concept of IC "fingerprints" to compare the characteristics of a suspected IC from a par-

978-3-9810801-3-1/DATE08 © 2008 EDAA

ticular family to the fingerprint of that family either in original form or in processed form. They apply power signals as the side channel and the processed power trace as the point of comparison. However, this method of trojan detection has its limitations in modern nano-scale technology based ICs, where the amount of parameter variation can be much more than $\pm 7.5\%$. Also, in more sophisticated ICs such as microprocessors, the amount of area that the trojan circuit occupies might be much smaller than 0.01%, and as a result, their effect on the measured physical quantity such as power or supply current might be too small to be detectable.

In [6], the authors propose "physical unclonable functions" for device authentication and secret key generation, which takes advantage of the random variations in an IC fabrication process. The key is generated from complex physical characteristics of each physical instance of an IC, and is thus extremely difficult to predict or extract. However, this approach is not completely suitable for trojan detection because it does not prevent the insertion of trojan circuitry in the IC and causes unacceptable hardware overhead to obtain good levels of security.

In [7], a digital design flow based on the use of "Wave Dynamic Differential Logic" which have a reduced current signature at the cost of more area overhead. This does not prevent the insertion or detection of additional trojan circuitry in the design.

3. Trojan Examples and Taxonomy

A trojan circuit must be hard to detect during chip testing and normal use. The basic model of a trojan circuit can be broken down into two major components: (a) Triggering and (b) Payload activation logic. The triggering logic monitors a set of q-external inputs (q-trigger) in order to activate the payload at the proper event. Triggering should only occur under very rare conditions. This is a critical property to minimize detection. To achieve minimal activation, the hacker could make the trojan triggering conditionally dependent on nodes with low controllability. The hacker is also likely to make the trojan affect nodes which are less observable (e.g. the write enable signal for the memory, set/reset signals for the state elements, etc.). Similarly, from the designer's point of view, any design technique targeted towards detecting the trojan circuit should be able to control the otherwise low-controllable nodes and be able to observe the low-observable nodes easily. In the next section we discuss a technique for trojan test detection based on a low frequency triggering model.

In order to evade detection during chip testing, for example, the triggering logic can exploit the test enable (TE) control line to disable the trojan. This is why scan-based designs cannot improve the security and functional testing must be involved.

Once triggering has occurred, the trojan becomes activated and delivers a payload to p-external circuit nodes (p-Payload). This payload can be either destructive or non-destructive. A destructive situation would be to enable the memory write signal (WE) unexpectedly and thereby overwrite an existing value with a random value as shown in Figure 1. A non-destructive case would be to enable superuser privilege mode while in normal user mode. This is not directly destructive to the chip's basic operation, but it is at the software operating system level.

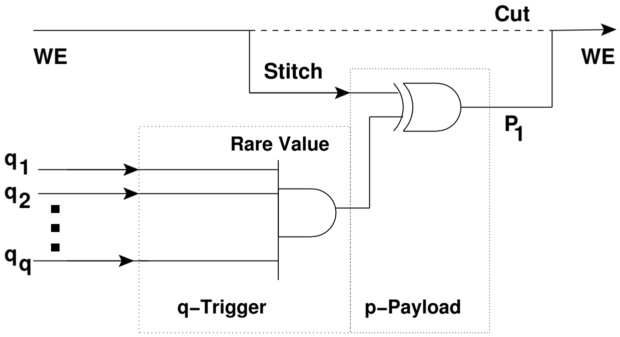
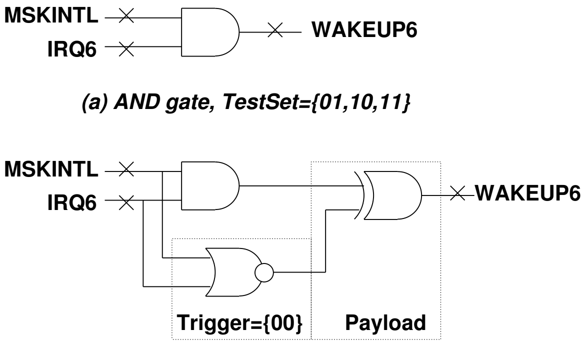


Figure 1. A trojan model

Even without a TE signal, test set vectors are not a sufficient condition in discovering trojan circuits. Figure 2a shows the designer's intention of waking-up a powered down circuit when a interrupt occurs. The single stuck fault (SSF) test set generated for the AND gate only covers three cases of the truth table's logic. The gaps in the test vectors are opportunities for hacking. In Figure 2b, the hacked circuit of 2a, uses the missing test case of $\{00\}$ to modify the truth table of the AND gate. The activation of the payload results in consuming excessive battery energy by preventing the circuit from going to sleep mode.



(a) AND gate, TestSet={01,10,11}

(b) Hacked Circuit, Passes s–a–f test

Figure 2. Trojan Circuit evading Single Stuck Fault testing

One might consider exhaustive testing by using 2^{n+m} test vectors for n-inputs and m-flipflops might be sufficient for detecting trojan circuits. Figure 3 shows a Time Bomb as a counterexample to using exhaustive testing. Neither the Designer nor the Tester is aware that extra state-machine logic has been slipped in between an error status signal (ER). The k-bit counter is always set longer than the test application time. The tester has no way of knowing what k is without physically destroying the chip. The Time Bomb increases the number of exhaustive test vectors from 2^{n+m} to 2^{n+m+k} where k is unknown. Thus, for a combinational logic only trojan, the detection problem is decidable but NP-complete. For a state-machine based trojan circuit, the detection problem is undecidable. An extension of the rare value is a rare sequence of values in which a particular ordered sequence of rare values triggers the trojan circuit. Furthermore, each of these sequences can be delayed by using a time bomb between the sequences.

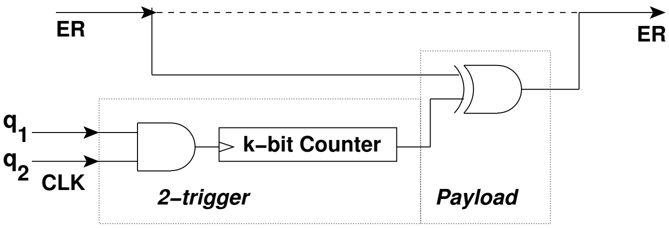


Figure 3. The Time Bomb

In general, trojan circuits can be classified by the nature of their trigger and payload mechanisms, as shown in Fig. 4. The trigger mechanism can be digital or analog based. An example of a mixed analog-digital trigger consists of using an on-die temperature sensor which triggers at a high temperature in conjunction with a parity error. Furthermore, a trigger can be delayed by a k-bit synchronous or asynchronous counter. Asynchronous designs are more difficult to detect, consuming power only when they change state.

The payload mechanism can also be digital or analog based. Digital payloads consist of altering a control, status, or data line. Analog payloads can result in increasing toggling activity resulting in faster battery drains in low poor designs. Other payloads, such as increasing the signal delay, or inserting bridging faults. Only rare value digital trojan circuits are considered in this paper.

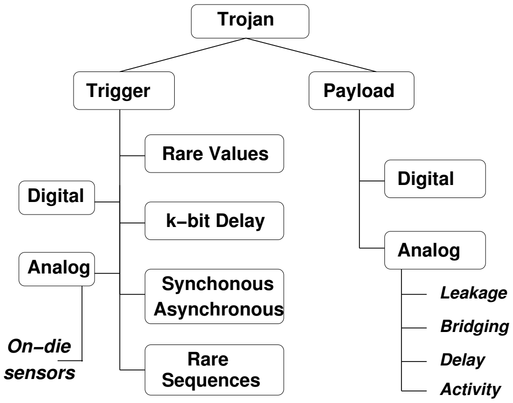


Figure 4. Trojan Circuit Taxonomy

4. Trojan Detection Approach

In this section we discuss trojan models and procedures to generate trojan detection vectors. Our trojan circuit model is shown in Fig. 5. Although this trojan has 2 inputs and one output, generally a trojan may have $q \geq 1$ trigger inputs $q_1, q_2, \cdots$ and $p \geq 1$ payload outputs $p_1, p_2, \cdots$. Without loss of generality, we will use the 2-input 1-payload output trojan in Fig. 5 to describe our method. Insertion of a trojan in a circuit entails two actions: a) *attach*, i.e. connecting the trojan inputs to circuit edges (wires); b) *stitch*, i.e. breaking one edge then feeding the left end to the trojan input and the right end to the trojan output. Note if there is more than one payload then a corresponding number of circuit edges would be stitched.

The trojan operates in two modes, *normal* and *trigger*. Normally, the trojan monitors the trigger inputs, while maintaining a stitch path, in lieu of the broken path RP, going through the trojan. This has no effect to the normal circuit operation, i.e. the bit values P and R are equal. Then, the stitch path is "triggered" producing a different payload, meaning the values P and R are not equal, e.g. $P = R'$. This is an error that may propagate as a fault to the circuit output. The key notion of the trojan threat is that the trigger values would occur very rarely to avoid detection during regular ATPG testing. By trigger frequency $f(v)$ we mean the frequency of occurrence of the trigger value v under all possible vectors at the circuit input. Then $f(v)$ should be very small, possibly 1 but $f(v) > 1$. Note that exhaustive testing will eventually produce the trigger value and hence detect the trojan as circuit stuck-at-fault (s-a-f). However, ATPG testing may not detect the trojan.

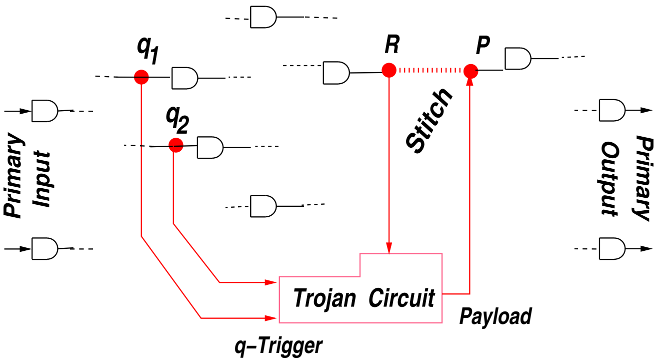


Figure 5. Trojan circuit model

A *trigger vector* is a primary input vector that triggers a trojan. For a trojan to be effective, in addition to triggering, the payload error must propagate to the circuit output. A *trojan test vector* is a trigger vector that propagates the payload to the circuit output. The set of trojan test vectors is a subset of the set of trigger vectors.

The problem is to find trojan test vectors that can detect all trojan effects triggered by rare values. Note, trojans triggered by "non rare" values should be detected by ATPG testing. Although there are numerous insertion places of trojans in a circuit, we are interested in most likely target sites to attach and stitch a trojan. The rules are, assuming trojans with q inputs and a threshold of f_{th} trigger frequency: rule 1) attach target are all combinations of q edges that attain one or more bit-values with frequency $\leq f_{th}$; rule 2) stitch targets for payload are all edges having low probability of fault propagation to the circuit output – in other words, low observability of the stitched edge with respect to the output. The rationale for rule 2 is to make the trojan less visible to ATPG tools.

These rules 1 and 2 are the basis of two procedures that we propose to handle the trojan threat. The procedures use a logic and fault simulator, and an ATPG tool. First, we developed the trojan target analysis procedure to identify all target sites of a given circuit for q-input trojan insertion To implement rule 1, the logic simulator accumulates q-bit values occurring on combinations of $q \geq 1$ edges for frequency thresholds, $f_{th} = 0,\ 1,\ 2,\ 3, \cdots$. The procedure collects all target sites to attach a q-input trojan with their respective trigger values and frequencies. Moreover, the procedure produces for each trigger value its corresponding input trigger vectors of circuit. To implement rule 2, we use a fault simulator to identify low observability nodes in the circuit that are candidates for stitching.

At the end of the trojan target analysis procedure we have: 1) a set of $\mathbf{Q}$ targets to attach q-input trojans, 2) a set $\mathbf{P}$ of target edges to stitch q-input trojans, 3) the trigger values and frequencies of each of the $\mathbf{Q} \times \mathbf{P}$ possible trojan circuits, 4) the input trigger vectors associated with the trigger values for the trojans.

The second procedure (trojan detection) produces trojan test vector sets to detect trojans in a circuit. The trojan detection procedure uses an ATPG tool to check whether each input trigger vector from the target analysis procedure can be propagated to the circuit output. This way, the set of input trigger vectors can be further compacted into the smaller set of trojan triggered vectors.

Note that the target trojan analysis procedure above assumed exhaustive exercising of all 2^n input patterns where n is the number of circuit inputs. However, if n is large, we may avoid this difficulty by using a suitable pseudorandom set of input patterns to perform the trojan target analysis.

5. Results

Circuit	PI	ATPG vectors	Trigger coverage	Trigger vectors	Escape vectors
c17	5	5	100%	0	2
c432	36	43	0%	6	19
c880	60	36	80.97%	56	38
c1355	41	85	58.54%	135	4
c5315	178	77	71.92%	301	232
c6288	32	29	27.48%	283	148
c7522	207	99	84.03%	602	230

Table 1. Results on Benchmark Circuits

The trojan detection procedures have been implemented and incorporated in a trojan tool set that includes a logic and fault simulator and a testability analysis tool. Our trojan tool set works together with a commercial ATPG tool (Tetramax from Synopsys). To validate our methods, we experimented with a number of circuits from the ISCAS85 benchmarks.

Our results are shown in Table 1. Column two is the number of primary inputs for each of the ISCAS85 circuits described in first column. Column three is the number of ATPG vectors that were generated by Synopsys Tetramax for single stuck-at faults for each of the circuits prior to any trojan analysis. Column four shows the Tetramax trigger vector coverage percentage of column three for $q = 2$ and trigger frequency threshold of one (i.e. $f_{th} = 1$). Thus, trigger vector coverage is the subset of Tetramax vectors in column three which trigger a trojan divided by the the total trigger vectors in column five. This column emphasizes that as expected the stuck-at coverage is not sufficient for trigger coverage. The circuit c17 is a trivial circuit which in fact had no rare values for threshold one, thus the 100% coverage. Column five is the number of primary input trigger vectors which trigger the trojan based on random sampling. The last column is the number of escape vectors which is the number of addition vectors needed to cover the next threshold level of two. This gives insight as to how much more effort is required to reach the next threshold level. However, increasing the threshold level, makes the trojan more likely to be detected by ATPG. Threshold one is the most desirable.

6. Conclusions and Future Work

In order to detect any trojan instance by observing the output logic value, we must trigger it. Thus, generation of an optimal set of test vectors, which can trigger the trojans will remain as an important problem. We have presented comprehensive analysis of the problem and classified trojan attacks in several broad categories. We have also considered detection of trojan instances in a circuit using a test generation approach. Simulation results show that such a technique can be effective to detect most small combinational trojans, which can easily evade indirect detection techniques such as the one using a power signature.

References

[1] D. Agrawal and et al. Trojan detection using ic fingerprinting. *IEEE Symp. On Security and Privacy*, pages 296–310, 2007.

[2] D. Hely, F. Bancel, M. Flottes, and B. Rouzeyre. Secure scan techniques: a comparison. In *12th IEEE International On-Line Testing Symposium (IOLTS)*, pages 119–124, 2006.

[3] J. Kumagai. Chip detectives. *IEEE Spectrum*, 37(11):43–49, Nov. 2000.

[4] S. Paul, R. S. Chakraborty, and S. Bhunia. Vim-scan: A low overhead scan design approach for protection of secret key in scan-based secure chips. *VLSI Test Symposium (VTS)*, 2007.

[5] S. Ravi, A. Raghunathan, and S. Chakradhar. Tamper resistance mechanisms for secure embedded systems. *17th International Conference on VLSI Design*, 94(2):605 – 611, Feb. 2004.

[6] G. E. Suh and S. Devadas. Physical unclonable functions for device authentication and secret key generation. *Proceedings of the Design Automation Conference (DAC)*, pages 9–14, 2007.

[7] K. Tiri and I. Verbauwhede. A digital design flow for secure integrated circuits. *IEEE TCAD*, pages 1197–1208, 2006.