Security against Hardware Trojan through a Novel Application of Design Obfuscation

Rajat Subhra Chakraborty
Dept. of Electrical Engg. and Comp. Science
Case Western Reserve University
Cleveland, OH-44106, USA
rsc22@case.edu

Swarup Bhunia
Dept. of Electrical Engg. and Comp. Science
Case Western Reserve University
Cleveland, OH-44106, USA
skb21@case.edu

ABSTRACT

Malicious hardware Trojan circuitry inserted in safety-critical applications is a major threat to national security. In this work, we propose a novel application of a key–based obfuscation technique to achieve security against hardware Trojans. The obfuscation scheme is based on modifying the state transition function of a given circuit by expanding its reachable state space and enabling it to operate in two distinct modes – the *normal mode* and the *obfuscated mode*. Such a modification obfuscates the rareness of the internal circuit nodes, thus making it difficult for an adversary to insert hard-to-detect Trojans. It also makes some inserted Trojans benign by making them activate only in the obfuscated mode. The combined effect leads to higher Trojan detectability and higher level of protection against such attack. Simulation results for a set of benchmark circuits show that the scheme is capable of achieving high levels of security at modest design overhead.

Categories and Subject Descriptors

B.6.1 [**Logic Design**]: Design Styles—*sequential circuits*; K.6.5 [**Management of Computing and Information Systems**]: Security and Protection—*physical security*

General Terms

Design, Security

Keywords

Design obfuscation, hardware security, hardware Trojan

1. INTRODUCTION

The issue of *Trust* is an emerging problem in semiconductor integrated circuit (IC) security [1],[2]. A design can be tampered in an untrusted fabrication facility by the insertion of malicious circuitry (*hardware Trojan*) that triggers a malfunction under rare circuit conditions [5]. Due to the stealthy nature of hardware Trojans and inordinately large number of possible Trojan instances an adversary can exploit, detection of hardware Trojan by traditional post–manufacturing testing is extremely challenging [4]. Many existing approaches of Trojan detection rely on the measurement of *side–channel parameters* such as delay and power signature [2], [6]–[8]. However, they can be very susceptible to measurement and process–variation induced noise. Moreover, they suffer from reduced detection sensitivity in case of ultra–small Trojans consisting of few logic gates [2].

Obfuscation is a technique that transforms an application or a design into one that is functionally equivalent to the original but is significantly more difficult to reverse engineer [3]. In this work, we propose a novel application of design obfuscation in achieving security against hardware Trojans, at low design overhead. The obfuscation scheme is realized by modification of the state transition function that enables circuit operation in two distinct modes – (a) the *obfuscated mode* when circuit functionality is significantly different from the normal functionality, and (b) the *normal mode*, when its behavior is identical to its non–obfuscated version. The mode control is performed by the application of a specific sequence of input vectors on initialization, called an *initialization key sequence*. As a result of the obfuscation, the inserted Trojans either become *more detectable*, or *decrease in potency* by activating only in the *obfuscated mode*. In Section 2 we describe the obfuscation methodology for achieving protection against hardware Trojan. In Section 3 we provide simulation results to estimate the level of security due to obfuscation. We conclude in Section 4.

2. METHODOLOGY

2.1 State Transition Graph modification

The obfuscation in our scheme is achieved by three important modifications of the State Transition Graph (STG) of the circuit:

1. A single arc connects the sets of nodes of the STG corresponding to the *normal mode* and the *obfuscated mode* state space, i.e., there is only a single input pattern from a specific *obfuscated mode* state that can take the circuit to its *normal mode*.

2. The size of the reachable state–space is "blown up" by a large (exponential) factor using extra inserted state elements, and,

3. Certain states, which were *unreachable* in the original design are used and made reachable only in the *obfuscated mode* of operation.

ICCAD'09, November 2–5, 2009, San Jose, California, USA.

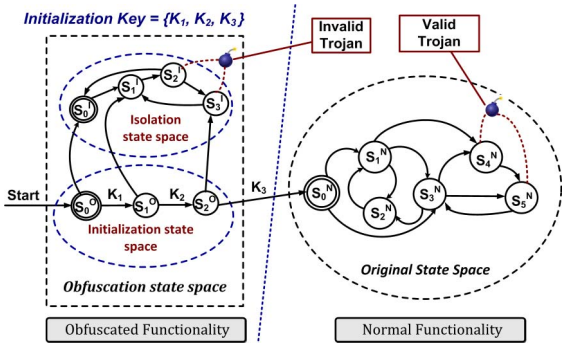

Figure 1: The obfuscation scheme for protection against hardware Trojans: modification of the STG hides the true rare events in a circuit and makes many possible Trojans invalid.

To design a hard-to-detect Trojan circuit, an intelligent adversary would typically find the rare events in a circuit i.e. the node-value pairs which are difficult to control and observe, in order to construct Trojan trigger conditions and payloads [4]. To find the rare trigger conditions, an adversary would require accurate estimations of internal node signal probability. This can be efficiently done by multiple random initialization of a given circuit to a reachable state and then applying random input vectors [9]. However, if the starting state in the adversary's simulations is in the *obfuscated mode*, because of the extreme rareness of the condition that allows the transition from *obfuscated* to *normal mode*, the simulations would most likely remain confined in the *obfuscated mode*. As a result, the signal probability for the circuit nodes calculated by the adversary would deviate significantly from those calculated if the simulations would have taken place with the states in the *normal mode*. Similar situation would prevent the adversary from finding the poorly observable nodes as potential payloads of a Trojan. Hence, if the adversary designs and inserts a Trojan based on wrong controllability/observability, there is a high probability that the Trojan would be triggered and detected with post–manufacturing logic testing. To increase this probability, the size of the *obfuscation state space* should be made as large as possible compared to the *normal state space*, by the addition of n extra state elements.

Moreover, a Trojan instance may depend on one or more states in the *obfuscated mode* to either trigger or propagate its malicious effect to the primary output of the circuit. Since the *obfuscation state space* is unreachable in the *normal mode* of operation, the inserted Trojan, whose activation and/or observation condition depend on the obfuscation state space, would not activate or have its effect propagate to the primary output or state elements during normal operation. Such Trojans would thus be rendered functionally *benign* or invalid.

An example STG of an obfuscated circuit is shown in Fig. 1. The obfuscation principle is similar to the one proposed in [3]. After power up, on the application of an input sequence $K_1 \rightarrow K_2 \rightarrow K_3$ during a process of *initialization*, the circuit reaches the state S_0^N, which is the starting state in the *normal state space*, allowing *normal mode* of operation. The set of states that the circuit goes through during the initialization process constitutes its *initialization state space*. On the application of even a single incorrect key during initialization, the circuit goes to a set of states which we refer as the *isolation state space*, from which it is not possible to come back to the *initialization* or the *normal* state space. The *initialization state space* and the *isolation state space* both can be realized by using additional state elements (SEs) to the design. Together they constitute the *obfuscation state space*. Note that due to the exponential dependence between the number of state elements and the size of the state space, a few additional state elements would help to greatly increase the size of the obfuscation state space.

The modifications of the STG are implemented in RTL and integrated with the existing gate-level description. The modified description is then re-synthesized using a logic synthesis tool under the specified design constraints. The logic optimization during this process *flattens* the netlist and hides the modifications made to internal circuit nodes. Derivation of the STG of a sequential circuit from a netlist is a *NP-complete* problem [7], and thus computationally infeasible for large circuits. This makes reverse-engineering of the obfuscation scheme by structural analysis almost infeasible for an adversary.

The construction of the *obfuscation state space* can significantly benefit from the determination of *unreachable states* in a given circuit, since these states can be used either in the initialization or isolation state space. Selection of unreachable states can be performed as below. First, a set of S state elements are selected at random from a given netlist and all the possible 2^S states are enumerated. Then, each of these 2^S states are subjected to *full sequential justification* at the inputs of the selected S state elements, using a justification tool such as *Synopsys Tetramax*. The justified states

Algorithm 1 Procedure *OBFUSCATE*
Generate the obfuscated netlist from a given circuit netlist

Inputs: Circuit netlist, maximum area overhead (*max_area_overhead*), number of states in the *obfuscation state space*, length of *initialization key sequence* (k)
Outputs: Obfuscated circuit netlist, *initialization key sequence*

1: Guess number of extra state elements to be added (n)
2: Guess number of original state elements to be used for state encoding (S)
3: **repeat**
4: **repeat**
5: Select S state elements randomly from circuit
6: Determine unreachable states for S state elements using sequential justification
7: **until** sufficient unreachable states found
8: Generate state encodings for the extra state elements
9: Generate random state transitions for the extra state elements
10: Generate random *initialization key sequence* of length k
11: Generate RTL for *obfuscation state space*
12: Integrate generated RTL with existing netlist
13: Re–synthesize modified circuit
14: Calculate *area_overhead*
15: $n \Leftarrow n-1$, $S \Leftarrow S-1$
16: **until** $area\_overhead \leq max\_area\_overhead$

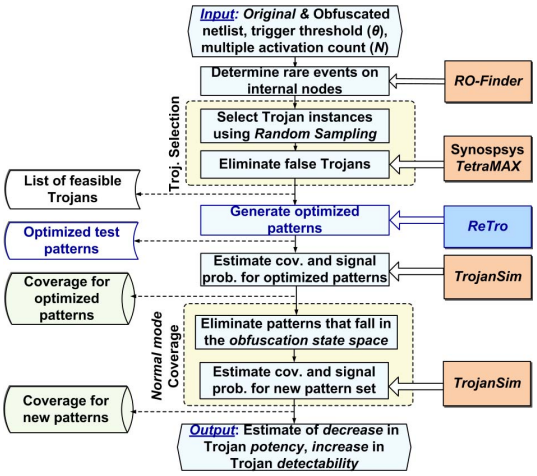

Figure 2: The framework to evaluate the effectiveness of the obfuscation scheme.

are discarded, while the states which fail justification are collected to form the set of structurally unreachable states. Random state encodings are generated for the n extra state elements. Algorithm 1 shows the major steps of the procedure ***OBFUSCATE*** that performs obfuscation of a circuit for a given area overhead constraint.

2.2 Test generation for Trojan detection

We noted through simulations on a number of benchmark circuits that a representative sample of Trojan instances (10–20K) from a combinatorially large population can be selected that reliably estimates the coverage metric for a given test set. First, the signal probability at the internal nodes of the circuit are estimated by the application of a large set of random vectors to the circuit. From the signal probability (S_p) of the internal nodes, we select a set of candidate trigger nodes with S_p less than a specified trigger threshold (θ). Next, starting from a large set of *weighted random vectors*, a smaller testset is generated to excite each of these candidate trigger nodes to its rare value at least N times, where N is a given parameter [4]. Note that for sequential circuits, we assume a *full–scan* implementation. Sequential justification is applied to eliminate *false Trojans*, i.e. Trojans which cannot be triggered during the operation of the circuit.

2.3 Automated design flow

The procedure in Algorithm 1 can be easily automated. We implemented the steps as well as a scheme to evaluate the effectiveness of the approach. Fig 2 shows the evaluation framework that integrates a number of tools interfaced by TCL scripts. Fig. 2 shows the steps to estimate the effectiveness of the obfuscation scheme. The computation of signal probabilities at the internal nodes is done by the C program *RO–Finder* (**Rare O**ccurrence **Finder**) [4]. We performed random sampling from a set of four-input *combinational Trojans*. The testset for Trojan detection based on multiple excitation of rare trigger conditions is performed by the C program *ReTro* (**Re**duced pattern generator for **Tro**jans) [4]. The generation of the reduced pattern set by eliminating the patterns with states in the *obfuscation state space* is performed by a TCL program. The decrease in the Trojan *potency* and the increase in the Trojan *detectability* are then estimated by a cycle-accurate simulation of the circuit by the simulator *TrojanSim* (**Trojan Sim**ulator) [4] written in C. *TetraMax* is used for sequential justification of the Trojan triggering conditions. Fig. 2 shows the steps to estimate the effectiveness of the obfuscation scheme. All simulations were carried out on a Hewlett–Packard Linux workstation with a 2GHz dual–core processor and 2GB main memory, using a LEDA 250nm library.

The original design is subjected to a large set of random vectors, while a modified set of random vectors which ensure operation only in the *normal mode* is applied to the obfuscated design. The increase in Trojan *detectability* is estimated by comparing the respective coverage values obtained for the obfuscated and the original design for the same number of test patterns. To determine the decrease in *potency* of the Trojans by the proposed scheme, from a given vector set we eliminate those vectors with state values in the *obfuscation state space*. We then re–simulate the circuit with the reduced test set to determine the Trojan coverage. The decrease in the Trojan coverage obtained from the reduced test set indicates the Trojans which are activated or effective only in the *obfuscation state space* and, hence, are effectively benign.

3. RESULTS

Fig. 3 shows the variation in the percentage of Trojans rendered benign, percentage of internal nodes with false signal probability, and the percentage increase in detectability of Trojans for the *s1196* ISCAS-89 benchmark circuit. Table 1 shows the effects of obfuscation on increasing the security against hardware Trojans for a set of ISCAS–89 benchmark circuits with 20,000 random instances of four-input Trojans, for a trigger threshold (θ) of 0.2 and N=1000. The length of the initialization key sequence was four for all the benchmarks. From these plots and Table 1 it is evident that the construction of the *obfuscation state space* with even a relatively small number of state elements still makes a significant fraction of the Trojans benign. Moreover, it obfuscates the true signal probabilities of a large number of nodes.

Fig. 4 shows for four-input Trojans the two different effects by which Trojans are rendered benign - i.e. some of

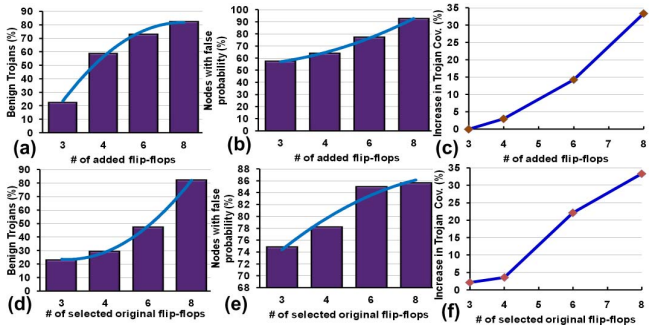

Figure 3: Level of protection against Trojans in benchmark circuit *s1196* as a function of - (a), (b), (c): the number of added flip–flops in state encoding (n) with $S = 4$, and (d), (e), (f): the number of original state elements used in state encoding (S) with $n = 4$.

Table 1: Effect of Obfuscation on Security Against Trojans (100,000 random patterns, 20,000 four input Trojan instances, $\theta = 0.2$)

Ckt.	# Troj. Inst.	Obfus. Flops (n + S)	Obfuscation Effects (%)		
			Benign Troj.	False Prob. Nodes	Troj. Cov. Incr.
s1488	192	98	60.53	71.02	12.12
s5378	2641	331	70.28	85.05	15.00
s9234	747	20	62.50	65.62	25.00
s13207	1190	36	80.77	83.59	20.00
s15850	1452	124	77.78	79.58	18.75
s38584	342	11	71.43	77.21	50.00
Avg.	**1094**	**≈103**	**70.55**	**77.01**	**23.48**

them are triggered only in the *obfuscation state space*, while the effect of some are observed (i.e. propagated to primary output or SE input) only in the *obfuscation state space*. Fig. 5 shows the improvement in Trojan detection coverage in the obfuscated design compared to the original design for the same number of random vectors and a random population of four-input Trojans. This plot illustrates the net effect of the proposed obfuscation scheme in increasing the level of protection with an average increase of 20.24% in the Trojan detection coverage.

Table 2 shows the design overheads (at iso–delay) and the run–time for the proposed obfuscation scheme. The proposed scheme incurs modest area and power overheads, and the design overhead diminishes with increasing size of the circuit. The run–time presented in the table is dominated by the sequential justification performed by *TetraMax*, which takes about 90% of the total time.

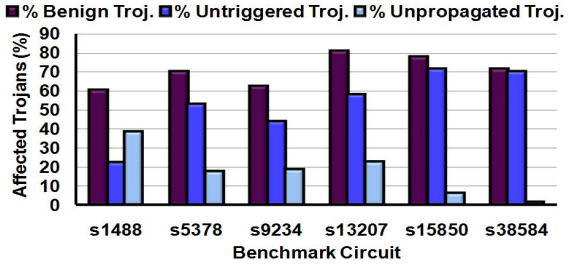

Figure 4: Effect of obfuscation on Trojans with four trigger nodes.

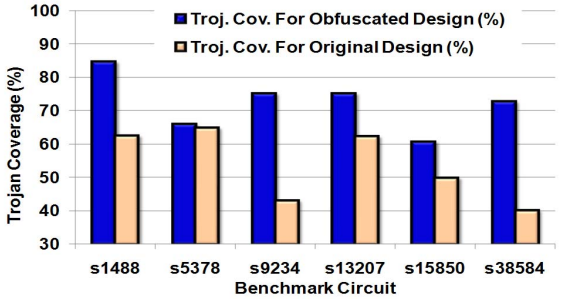

Figure 5: Improvement of Trojan coverage in obfuscated design compared to the original design for Trojans with four trigger nodes.

Table 2: Design Overhead (at iso–delay) and Run–time for the Proposed Design Flow

Benchmark Circuit	Overhead (%)		Run–time (mins.)
	Area	Power	
s1488	20.09	12.58	31
s5378	13.13	17.66	186
s9234	11.84	15.11	1814
s13207	8.10	10.87	1041
s15850	7.04	9.22	1214
s38584	6.93	2.63	2769
Average	**11.19**	**11.34**	**1175.83**

4. CONCLUSIONS

Malicious modification of integrated circuits in untrusted fabrication facilities has emerged as a serious security threat. Conventional logic test generation and application techniques cannot be readily extended to reliably detect hardware Trojans. We have presented a novel application of design obfuscation to achieve effective protection against hardware Trojans. Obfuscation of a circuit can make it difficult for an adversary to interpret its operation, resulting in Trojan insertion that either becomes functionally benign, or easily detectable by logic testing. The required design modifications can be easily automated and integrated with the conventional design flow. Simulation results for a set of sequential benchmark circuits show that a well–formulated obfuscation scheme can provide enhanced protection against hardware Trojans at low design overhead.

5. REFERENCES

[1] S. Adee. The hunt for the kill switch. *IEEE Spectrum*, 45(5):34–39, May 2008.

[2] D. Agrawal, S. Baktyr, D. Karakoyunlu, P. Rohatgi, and B. Sunar. Trojan detection using IC fingerprinting. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 296–310. 2007.

[3] R. S. Chakraborty and S. Bhunia. Hardware protection and authentication through netlist level obfuscation. In *Proceedings of IEEE/ACM ICCAD*, pages 674–677. 2008.

[4] R. S. Chakraborty, F. Wolff, S. Paul, C. Papachristou, and S. Bhunia. MERO: A statistical approach for hardware trojan detection. *Lecture Notes in Computer Science*, 5747:396–410, September 2009.

[5] D. Collins. Trojan detection using IC fingerprinting. `http://www.darpa.mil/BAA/BAA06--40mod1.html`, 2006.

[6] Y. Jin and Y. Makris. Hardware protection and authentication through netlist level obfuscation. In *Proceedings of IEEE HOST*, pages 51–57. 2008.

[7] A. L. Oliveira. Robust techniques for watermarking sequential circuit designs. In *Proceedings of ACM/IEEE DAC*, pages 837–842. 1999.

[8] R. M. Rad, X. Wang, M. Tehranipoor, and J. Plusquellic. Power supply signal calibration techniques for improving detection resolution to hardware Trojans. In *Proceedings of IEEE/ACM ICCAD*, pages 632–639. 2008.

[9] M. G. Xakellis and F. N. Najm. Statistical estimation of the switching activity in digital circuits. In *Proceedings of ACM/IEEE DAC*, pages 728–733. 1994.