2008 11th IEEE High Assurance Systems Engineering Symposium

Randomization Based Probabilistic Approach to Detect Trojan Circuits

Susmit Jha
EECS, UC Berkeley

Sumit Kumar Jha
School of Computer Science, CMU

Abstract—**In this paper, we propose a randomization based technique to verify whether a manufactured chip conforms to its design or is infected by any *trojan* circuit. A trojan circuit can be inserted into the design or fabrication mask by a malicious manufacturer such that it monitors for a specific rare *trigger* condition, and then it produces a *payload* error in the circuit which alters the functionality of the circuit often causing a catastrophic crash of the system where the chip was being used. Since trojans are activated by rare input patterns, they are stealthy by nature and are difficult to detect through conventional techniques of functional testing. In this paper, we propose a novel randomized approach to probabilistically compare the functionality of the implemented circuit with the design of the circuit. Using hypothesis testing, we provide quantitative guarantees when our algorithm reports that there is no trojan in the implemented circuit. This allows us to trade runtime for accuracy. The technique is sound, that is, it reports presence of a trojan only if the implemented circuit is actually infected. If our algorithm finds that the implemented circuit is infected with a trojan, it also reports a *fingerprint* input pattern to distinguish the implemented circuit from the design. We illustrate the effectiveness of our technique on a set of infected and benign combinational circuits.**

I. Introduction

The issue of trust is gaining significance in the hardware industry. Motivations like higher efficiency and better quality have led to increased use of commercially manufactured micro-chips in high integrity systems like military facilities, communications and aerospace systems. Procuring chips from commercial suppliers spread across the globe allows continuous upgrading to rapidly changing state-of-the-art in chip design and fabrication. It has become commercially infeasible to secure the entire IC supply chain [1]. This has increased the concern about security of micro-chips from insertion of trojan circuits during manufacturing [2], [3]. Hence, it is pertinent to develop techniques which can be used to validate a micro-chip manufactured under untrusted environment.

A trojan circuit can be implemented by adding extra logic to the chip being fabricated, which could be done either during manufacturing or during the chip's design phase. A saboteur could modify one of the masks used to imprint the circuit layout onto the semiconductor wafer or the chip design could be mutated to add the extra logic. Common examples of such trojan circuits are kill switches or reprogramming instructions to FPGA-based architecture [4]. Since the extra logic added to the circuit is often 3-4 orders of magnitude smaller compared to the circuit itself [5], there is little structural difference between the circuit and its infected version. Also, the trojan circuits are activated through a rare trigger event. Thus, the functional difference between a circuit and its infected version would manifest itself only when a very particular input pattern is applied to a chip. These characteristics of the trojan circuits make it extremely stealthy. Thus, the trojans are very difficult to detect using any conventional testing technique which are developed to find physical and logical bugs in implementation [5], [6], [7].

Trojans can be intelligently built to deter any detection through vigilant approaches based on LFSR and Logis BIST [8], [9]. Destructive testing of a few chips does not guarantee that the other chips are not infected. A saboteur might deliberately insert trojans only in a fraction of all the chips manufactured in a batch to avoid being detected by any technique based on sampling and destructive testing. Further, the trojans remain passive for most of the time unless triggered and hence, are not usually observable through circuit characteristics like power, temperature and electro-magnetic profiles [6].

In this paper, we focus on detecting trojans in combinational circuits. Our technique can be applied to sequential circuits by unfolding it to a finite number of steps using bounded model checking [10]. Our technique first uses randomization arguments to construct unique probabilistic signature of a circuit. We find a probability distribution on the inputs such that the probability distribution of output is unique for every functionally distinct circuit. Then, we propose a technique based on hypothesis testing to statistically infer the presence of a trojan in circuit under test (CUT). The output of our technique is either an input pattern which distinguishes the functionality of the CUT from its design or a quantitative confidence level that the CUT has no trojans. The confidence level can be improved by running the analysis technique for a longer time.

The rest of the paper is organized as follows. In Section II, we discuss different techniques proposed in literature for detection of trojan circuits. We describe our approach to generate unique probabilistic signatures for functionally distinct circuits in Section III. In this section, we present theoretical results and develop instrument

1530-2059/08 $25.00 © 2008 IEEE
DOI 10.1109/HASE.2008.37

techniques to propose a trojan detection algorithm in the following section. In Section IV, we present the trojan detection algorithm and show how the algorithm can be made scalable by using signature hash which map functionally close circuit to the same signature. We show how hypothesis testing based statistical inference can be used to probabilistically check the presence of trojan circuits. In Section V, we present the results of experiments on a set of combinational circuits. We conclude by discussing ongoing work to extend this technique to test presence of trojans in sequential circuits and in software.

II. Related Work

The problem of trojan detection has been discussed only recently in literature. Two different approaches have been proposed to address this problem.

The first approach is invasive testing which is a destructive testing technique where each layer of a chip is inspected through X-ray tomography. This process involves demasking, delayering and layer-by-layer comparison of X-ray scans with the original mask. Apart from being very expensive, this approach depends on randomly sampling the chips and destructively testing these chips. There is no guarantee that the chips which were not tested and would be put in use do not have any trojan in them even if the circuits sampled for testing are found to be trojan free.

The second approach is based on side channel analysis, that is, using side channels like power to compare the corresponding characteristics of CUT and the original circuit. Agrawal et al [5] use power signals as side channel and detect presence of trojans by filtering the power signature of the trojan circuit from the noise. Their work has two key weaknesses. Firstly, they require that the power signature difference between the infected and benign circuit must exceed the process variation to be statistically significant and thus, to be filtered from noise. In modern nano-scale technology based ICs, the amount of parameter variation can be much more than 7.5%. Also, the trojan circuit area can be as low as 0.01% in which case the power signature difference would be too small [7]. Secondly, they use completely random inputs which may not be effective in attaining significant difference between the infected and benign circuit. Banga et al [6] build on the work in [5] and propose some heuristics to find set of inputs which would maximize the discrepancy in the power signatures and hence, help in detection of trojans. Their approach relies on reducing the power consumption of the CUT by switching it across states which are close in Hamming distance. This allows them to identify the high activity regions which are potential sites of trojan infections in the circuit. Their technique also can not detect trojans which hide their activity in the statistical noise.

The approach presented in this paper is the first functional testing based approach proposed for trojan detection to best of our knowledge. It is orthogonal to the techniques discussed above and can be used together with the above techniques to complement them in detection of trojans. While side channel analysis can provide a list of suspect chips, our functional differentiating technique can help identify inputs which trigger the trojan and hence, conclusively show that a chip is infected. Another key advantage of our technique is that unlike side channel cryptanalysis we do not depend on any measurement of physical phenomenon like power or current. Also, our validation technique is not destructive like invasive testing and hence, it is possible to test each chip before use.

III. Probabilistic Signature of a Circuit

In this section, we show how to obtain a probabilistic signature of a Boolean circuit. Without loss of generality, let us assume that the circuit has a set of inputs $\mathcal{I}$ and a single output O. For circuits with more than one output, we can consider each output and its fan-in cone as a separate circuit. We find a probability distribution, P, of the input assignment such that the random application of input from the distribution leads to a unique probability of the output of the circuit being 1. Then, distinguishing CUT with its design is done by finding the probability of the output being 1 by applying inputs randomly from the probability distribution P. If the CUT is infected with a trojan, then the probability of logic 1 at the output of CUT and original circuit will be different.

This idea of using randomization to distinguish functionally different circuit has been used in probabilistic equivalence checking [11], [12] of circuits. But there is a key difference between probabilistic equivalence checking of circuits and trojan detection. In traditional equivalence checking [13] both circuits - CUT and the original circuit are available as *white box*. Hence, it is possible to use techniques like satisfiability solving [14] and randomized structural hashing [15], [16]. In contrast, the problem of trojan detection is more complicated since we only have the original circuit design and the CUT is only available as a black box. Though we have the circuit design as a completely observable Boolean circuit, chip under test needs to be tested for equivalence with the design by observing only its input/output behavior.

We now develop a probabilistic technique to check equivalence of a black box implementation with respect to a whitebox design.

A. Characteristic Polynomial of a Circuit

We use lower case letters for Boolean variables x_i, $i = 1, 2, \ldots N$ taking value in $\{0, 1\}$. We use upper case letters for real variables X_i, $i = 1, 2, \ldots N$ taking values from $\mathbb{R}$. A Boolean function f is a function from $\{0, 1\}^n$ to $\{0, 1\}$. The corresponding polynomial $P(f)$ is a mapping from $\mathbb{R}^n$ to $\mathbb{R}$. It also must be noted that Boolean operators *and* and *or* are similar to the arithmetic operators *multiply* and *add* over the finite field $GF(2)$ respectively. Also, the Boolean operators are idempotent, that is,

- $f \wedge f = f$
- $f \vee f = f$

The definition of characteristic polynomial [12], [11] is based on these properties.

Definition 3.1: Characteristic Polynomial - Given a Boolean function f, the *characteristic polynomial* $P(f)$ is defined inductively as follows:

- $P(f(\ldots, x_i, \ldots) = (1 - X_i)P(f_{x_i=0}) + X_i P(f_{x_i=1})$
- $P(x_i) = X_i$
- $P(true) = 1$
- $P(false) = 0$

where $f_{x_i=0}$ and $f_{x_i=1}$ are co-factors of f with respect to x_i obtained by setting $x_i = 0$ and $x_i = 1$ respectively. The idempotent property can be enforced through *exponent suppression* that is all exponents in the polynomial are reduced to 1.

For the common binary functions, the characteristic polynomial would be as follows -

- $P(x \wedge y) = XY$
- $P(x \vee y) = X + Y - XY$
- $P(\neg x) = 1 - X$

The following examples illustrate the characteristic polynomials obtained from the Boolean formulae after exponent compression.

1) $P[(x \wedge y) \vee (x \wedge z)] = XY + XZ - X^2YZ = XY + XZ - XYZ$
2) $P[(x \vee x) \wedge y] = (X + X - X^2)Y = XY$
3) $P[(x \wedge y) \vee (x \wedge \overline{y})] = XY + X(1 - Y) = X$

It has been shown [11] that when the inputs are assigned randomly selected real numbers, the characteristic polynomials of two Boolean functions give the same value with probability 1 if and only if the functions are identical. This follows from the fact that there is a unique embedding of Boolean functions into a polynomial ring over any finite field such that they have the same value when all variables take values 0 or 1. Properties of characteristic polynomials are well-studied in literature [11], [12] The following theorem summarizes the main result used in our technique.

Theorem 3.2: Given two Boolean formulae f and g, $f = g$ if and only if the characteristic polynomial are identical, that is, $P(f) = P(g)$.

The characteristic polynomials of a Boolean formulae and the above uniqueness property provide an efficient way to test the correctness of any system by computing their characteristic polynomial of the design and implementation and matching whether the polynomials are identical. But such an use of characteristic polynomials in verification has two main impediments.

1) Both the design and implementation should be available as white-boxes, that is in a description that can be translated to a Boolean formulae so that the characteristic polynomial can be computed for both and matched. This is not possible in most applications where implementation is only partially observable. In case of detecting trojans in a chip, the implementation is a black-box and hence direct computation of polynomial representing implementation is not possible.
2) It has been shown by Kumar et al [11] that the computation complexity of computing a characteristic polynomial for a n -input circuit is $O(n2^n)$ and hence, it is not feasible for even circuits of moderate size.

The characteristic polynomial can also be interpreted probabilistically. The assignments from real domain $[0, 1]$ can be interpreted as probabilities of logic 1. Then, the corresponding value of the characteristic polynomial gives the probability of output logic 1 for the Boolean function. Thus, the probability of an output of a Boolean function can be calculated using the characteristic polynomial of the function. Hence, the characteristic polynomial of a Boolean function can also be called its *probability expression*. This probabilistic interpretation of the characteristic polynomial allows us to use statistical techniques to test whether a circuit is infected or not.

B. Probability Signature

In this section, we show that there exists a probability assignment to inputs such that the probability of output of the Boolean function is unique for each function. We identify a set of assignments in $\mathbb{R}$ such that each probability expression $P(f)$ for any Boolean function f evaluates to distinct values in $\mathbb{R}$. We generalize the similar assignments discussed in [17], [12] for probabilistic equivalence checking of white-box circuits.

We now state and prove the main result in this section.

Theorem 3.3: Let f, g be a Boolean functions with common inputs $\{x_0, x_1, \ldots, x_{n-1}\}$ and $P(f), P(g)$ be the probability expression with corresponding variables $\{X_0, X_1, \ldots, X_{n-1}\}$. For the variable assignment given by the following equation

$$X_i \longrightarrow \frac{p^{2^i}}{p^{2^i} + 1}$$

where p is an integer greater than 1, the characteristic polynomials $P(f)$ and $P(g)$ evaluate to the same value if and only if the Boolean functions f and g are equal.

Proof: Let us consider $m_j = k_{n-1} \ldots k_1 k_0$ which is binary encoding of j in n bits for $j = 0$ to $j = 2^n - 1$, that is, $j = \sum_{i=0}^{n-1}(k_i 2^i) = \sum_{k_i=1} 2^i$. Each m_j represents a particular assignment of 0 or 1 to the inputs of a n-input Boolean function such that the i-th bit of m_j is 1 if and only if $x_i = 1$.

Under the assignment, $X_i = \frac{p^{2^i}}{p^{2^i}+1}$, $X_i = 1 - \frac{p^{2^i}}{p^{2^i}+1} = \frac{1}{p^{2^i}+1}$

$$P(m_j) = (\prod_{X_{k_i}=1} \frac{p^{2^{k_i}}}{p^{2^{k_i}} + 1})(\prod_{X_{k_i}=0} \frac{1}{p^{2^{k_i}} + 1})$$

From elementary algebra, we know that
$(p^{2^k} + 1)(p^{2^k} - 1) = (p^{2^{k+1}} - 1)$
We use this to compute $P(m_j)$.
The denominator product is
$(p + 1)(p^2 + 1)(p^{2^2} + 1) \dots (p^{2^{n-1}} + 1)$
$= \frac{p-1}{(p-1)(p+1)(p^2+1)(p^{2^2}+1)\dots(p^{2^{n-1}}+1)}$
$= \frac{p-1}{p^{2^n}-1}$
The numerator product is $\prod_{X_{k_i=1}} p^{2^{k_i}} = p^j$ since $j = \sum_{X_{k_i=1}} 2^{k_i}$
Thus, $P(m_j) = \frac{p^j(p-1)}{p^{2^n}-1}$
$P(m_j) = C(p, n)p^j$ where $C(p, n)$ denotes $\frac{(p-1)}{p^{2^n}-1}$.

Any Boolean function can be expressed in sum-of-products form as the sum of its minterms. Let us assume that both f and g are presented as SOP of minterms. We denote the set of minterms in a Boolean function f by $M(f)$. Clearly, the sets $M(f)$ and $M(g)$ are equal if and only if f and g are equal.
Since, $f = \sum_{m_j \in M(f)} m_j$
$P(f) = \sum_{m_j \in M(f)} P(m_j)$ since $m_j \wedge m_i = false$ if $i \neq j$ and hence $P(m_j \wedge m_i) = 0$ for all i, j.
that is, $P(f) = \sum_{m_j \in M(f)} C(p, n)p^j$,
that is, $P(f) = C(p, n) \sum_{m_j \in M(f)} p^j$.
Intuitively, $\sum_{m_j \in M(f)} p^j$ is a number in p-ary number system and hence is unique for any given $M(f)$.
Hence, $P(f) = P(g)$, that is,
$C(p, n) \sum_{m_j \in M(f)} p^j = C(p, n) \sum_{m_j \in M(g)} p^j$
if and only if $M(f) = M(g)$, that is, $f = g$.
■

The above assignment of variables in $P(f)$ can also be treated as probability of assigning 1 to the inputs of the Boolean function f as discussed earlier. Thus, the following corollary follows from Theorem 3.3.

Corollary 3.4: Given two Boolean functions f and g over the n inputs $x_0, x_1, \dots, x_{n-1}$, if the probability of assigning x_i as 1 is $\frac{p^{2^i}}{p^{2^i}+1}$, then the probability of output of f and g being 1 is equal if and only f and g is equal.

Definition 3.5: The *probability signature* of a Boolean function f is the probability of output of f being 1 when the inputs are assigned using the probability distribution $p(x_i = 1) = \frac{p^{2^i}}{p^{2^i}+1}$.

Thus, probability signature is unique for each Boolean function. We illustrate this by tabulating the probability signature for all possible Boolean functions with number of inputs as 2. Any Boolean function over 2 inputs can be represented as a binary sequence of length 2^2 where the i-th element in the sequence represents the output of the function when the input is assigned $0, 0$ if i is 0; $0, 1$ if i is 1; $1, 0$ if i is 2 and $1, 1$ if i is 3. For example, the sequence 0001 denotes the binary AND gate which gives output 1 only when the input is 11. Similarly, the sequence 0111 denotes the binary OR gate. The following table shows the probability signature of all possible binary Boolean functions where the constant p is chosen to be 2.

TABLE I
Probability Signature of all Binary Boolean Functions

Function f	Signature P(f)	Function f	Signature P(f)
0000	0	0001	$\frac{8}{15}$
0010	$\frac{4}{15}$	0011	$\frac{12}{15}$
0100	$\frac{2}{15}$	0101	$\frac{10}{15}$
0110	$\frac{6}{15}$	0111	$\frac{14}{15}$
1000	$\frac{1}{15}$	1001	$\frac{9}{15}$
1010	$\frac{5}{15}$	1011	$\frac{13}{15}$
1100	$\frac{3}{15}$	1101	$\frac{11}{15}$
1110	$\frac{7}{15}$	1111	1

The direct determination of probability signature using evaluation of the probability signature is difficult since the computation complexity of obtaining probability expression is $O(n2^n)$ for a n input function [11]. Further, the accurate evaluation of the polynomial expression also requires high precision arithmetic. We discuss how to efficiently compute the probability signature using randomized testing followed by statistical analysis. We first present an information theoretic analysis of the signature to show that the most efficient signature would be obtained by setting $p = 2$ in the probability distribution of the input.

C. Information Theoretic Analysis of the Signature

We use Shannon's entropy to compare the space of signatures and the Boolean functions. Since we have an injective correspondence between the signatures and the Boolean functions, we require that the entropy of signature is atleast as much as the entropy of the Boolean function.

The entropy of a Boolean function f over n inputs is maximum if it is equally likely to be any Boolean function of n inputs. Hence, the maximum Shannon's entropy is given by

$$H(f) = \sum_i \frac{1}{2^{2^n}} \log 2^{2^n} = 2^n$$

This intuitively is in agreement with the fact that there are 2^{2^n} possible Boolean functions with n inputs and hence, any Boolean function with n inputs requires 2^n bits to be uniquely represented.

The entropy of probability signature is also maximum if each of the possible signature is equally likely. For any arbitrary p, Shannon's entropy is given by

$$H(P(f)) = \sum_i \frac{1}{p^{2^n}} \log p^{2^n} = 2^n logp$$

It follows that $\log p \geq 1$, that is $p \geq 2$. Choosing $p = 2$ is enough to ensure that one-to-one mapping exists between the signatures and the possible Boolean functions.

D. Probability Signature using Parameter Estimation

The probability signature of any circuit (Boolean function) is essentially the probability of getting 1 at the output of the circuit when the input vectors are selected randomly from the probability distribution given by $p(x_i = 1) = \frac{p^{2^i}}{p^{2^i}+1}$ where $p(x_i = 1)$ denotes the probability of applying 1 at the primary input x_i of the circuit. For a given circuit, with an unknown probability signature we can apply a number of random input vectors from this probability distribution and estimate the probability of the output being equal to 1.

Let the probability signature of the circuit be p. The random variable representing the output of the circuit being 1 when input vectors are chosen from the above distribution is clearly a Bernoulli variable which takes value 1 with success probability p.

If X is a random variable with Bernoulli distribution such that the expected value $E(X) = p$ and variance $var(X) = p(1 - p)$, then the *maximum likelihood estimator* of p from a random sample of size N is $\hat{p} = \frac{\sum_i X_i}{N}$

So, the maximum likelihood estimate of the probability signature of a circuit f can be obtained by applying inputs from the above probability distribution and by using the following simple formula $\hat{P}(f) = \frac{N_{f=1}}{N}$ where N is the total number of times the inputs where applied and $N_{f=1}$ is the number of times the output of f was observed to be 1.

IV. Trojan Detection Algorithm

In this section, we present the algorithm to detect trojans in a CUT using techniques developed in the last section. We first present an outline of the algorithm.

1) Compute probability signature of the design of the circuit using either its probability expression or through randomized testing followed by statistical estimation.
2) Use the probability distribution of input discussed above in Section III in forming the probability signature to generate random input vectors and apply it to the CUT and the design circuit.
3) If the output of CUT and design circuit differ for an input vector, then the algorithm terminates with the input vector as the *fingerprint* of the trojan infection.
4) After a set of N vectors have being applied and the CUT and design circuit gives the same output, we use statistical reasoning to give a *confidence interval* that they have the same probability signature and hence, the CUT is trojan free.

The first step follows immediately from the previous section. The probability signature of the circuit design can be obtained directly from the designer's high level description without having to construct the signature from the circuit synthesized from the high level description. For example, a circuit design in DSP applications like filters have already a polynomial representation and hence, can be directly used as the probability signature of the circuit. If the probability signature needs to be obtained from the circuit description, then we can use statistical estimate of the signature. This step is done offline and only once without the use of the fabricated chip and hence, is not the performance bottleneck of the technique.

The second step requires us to simulate the fabricated chip which is the CUT and the design circuit on a set of input vectors where the inputs are generated from the distribution used to define the probability signature. If the CUT is not infected by any trojan, then it is functionally identical to the design circuit and hence, its output must be same as the design circuit on each of the input vectors.

If an input vector is found at which the CUT and the design circuit differ, then the CUT is clearly functionally distinct from the design and is thus, infected by a trojan. Since, it is not possible to distinguish between an implementation error during fabrication and a deliberate trojan insertion, we assume that all functional difference correspond to malicious infection of the CUT. The input vector which illustrates the difference between CUT and the design is the *fingerprint* which is a proof of the trojan infection.

It is infeasible to test the CUT with n inputs for any significant fraction of the total number of possible 2^n input vectors. Hence, after applying $N << 2^n$ input vectors, if the output of the pattern as well as the design are identical, we use hypothesis testing to provide a *confidence interval* for the hypothesis that the CUT is free of any trojan infection.

Each simulation by application of test vector to the circuit and observation of the output is a Bernoulli trial with the output being 1 as the success event having probability p which is the probability signature of the CUT. The number of successes in Bernoulli trials is a random variable with Binomial distribution. If we count the number of times, the output was 1 in the simulation, then this will be a Binomial random variable. If the number of simulations N is fairly large, that is, $Np > 5$ and $N(1 - p) > 5$, then the Binomial distribution can be approximated as a normal distribution according to the *central limit theorem* [18]. Let us denote the corresponding normal variable as X.

Further, if $\hat{N}$ of the simulations resulted into output being set to 1, then $\hat{N}$ is a *point estimator* of X. We need to verify whether the random variable representing the outcome of the output on random simulations of the CUT has a mean which is same as the probability signature p. This can be done by verifying that $X \sim normal(Np, Np(1-p))$, that is, X is a random normal variable with mean Np and variance $Np(1-p)$. We verify this via Hypothesis Testing.

We need to test the following hypotheses where μ is the mean of the normal distribution of the random variable X.

H_0 : Mean of $\mu = Np$
H_1 : Mean of $\mu \neq Np$

Since X is a normal distribution, $Z = \frac{X-Np}{\sqrt{Np(1-p)}}$ is a standard normal distribution. Using the point estimator $\hat{N}$ of X, the test statistic is

$$Z_0 = \frac{\hat{N} - Np}{\sqrt{Np(1-p)}}$$

.

Given a significance level α, we will reject hypothesis H_0 if the test statistic $Z_0 > z_{\alpha/2}$ or $Z_0 < -z_{\alpha/2}$ where $z_{\alpha/2}$ is the $100\alpha/2$ percentage points of the standard normal distribution. We will fail to reject H_0 if $-z_{\alpha/2} \leq Z_0 \leq z_{\alpha/2}$. The *confidence* of the conclusion is given by $1 - \alpha$.

We explain this further with help of the following example. Suppose that, we are interested in accepting a CUT as trojan-free CUT with 95% confidence, then $\alpha = 0.05$. Further, let the probability signature be 7/15 corresponding to the 2-bit Boolean function represented as 1110 in Table I. Let $N = 10^5$ be the number of simulation vectors that were applied to the CUT and design-circuit, and the CUT did not differ from design in any simulation. Further, let the number of times the output was 1 be $45,016$. In order to declare the CUT, trojan free, we need to compute the test statistic Z_0 which is given by $\frac{45{,}016-7/15\times10^5}{\sqrt{10^5\times7/15\times8/15}} = -10.459$. Now, $z_{0.025} = 1.96$ and $z_{-0.025} = -1.96$. Since $Z_0 < -1.96$ in this case, we reject the hypothesis.

We can either terminate and output that the algorithm did not find a fingerprint but can not claim with 95% confidence that the CUT is trojan free or we can continue with experiments further upto $N = 2.10^5$ simulation vectors. If none of the 2.10^5 simulation vectors detect any difference in output of CUT and the design and we find that $\hat{N} = 93562$, we need to recompute the test statistic for rejection or acceptance of hypothesis. The test statistic Z_0 which is now given by $\frac{93562-7/15\times2.10^5}{\sqrt{2.10^5\times7/15\times8/15}} = -1.025$. Since $-1.96 \leq Z_0 \leq 1.96$ in this case, we are 95% confident that the CUT is free of trojans and we can terminate with this answer.

We now present some extensions of the main algorithm presented above. We first show how we can make this technique more scalable by using hash of probability signature instead of the signature itself. Then, we show how our technique can be applied to Boolean functions with multiple outputs. As intuitively obvious, more than one output provide more than one observation point in the circuit and hence, the detection of trojan infection in a multi-output circuit is easier compared to detection of trojans in a single-output circuit of same size. We also show that our technique would be able to detect trojans which cause greater functionality change more easily than trojans which cause smaller functionality change.

A. Probability Signature Hash

The probability signature matches each functionally distinct circuit to a unique signature. If the circuits are large, we can reduce the signature space by mapping more than one functionally distinct circuit to the same signature. We illustrate this with an example strategy here which reduces the signature space.

Instead of the probability distribution $p(x_i = 1) = \frac{p^{2^i}}{p^{2^i}+1}$ where $i = 0, 1, 2 \ldots (n-1)$ which creates unique signatures for circuits with n inputs, we consider the following hashed probability distribution

$$p(x_{2m+1} = 1) = p(x_{2m} = 1) = \frac{p^{2^m}}{p^{2^m} + 1}$$

for $m = 0, 1, 2, \ldots (n-2)/2$.

Using the same argument as in the proof of Theorem 3.3, it can be shown that each min-term will now be mapped to $C(p, n)p^j$ such that j is now an $n/2$-bit binary number. So, $\frac{2^n}{2^{n/2}} = 2^{n/2}$ minterms will be mapped to the same signature. Any function f and g will now be assigned a signature corresponding to $C(p, n) \sum_{m_j \in M(f)} p^j$ where the sum term is a number in the p-ary number system. For each bit of this p-ary number, any one of the $2^{n/2}$ minterms corresponding to p^j could be present. Thus, 2^{2^n} possible functionally distinct circuits of input length n are assigned signatures of length $p^{2^{n/2}}$ and hence, each signature corresponds to $\frac{2^{2^n}}{p^{2^{n/2}}}$ distinct Boolean functions. For $p = 2$, this collision set is $2^{2^{n/2}}$.

This hashing technique is of use when we know the set of inputs of the function which are most expected to be used by the trojan circuit. The probability distribution of the input determines the *activity* of the input, that is, how frequently the input is toggled while simulation and hence, a higher activity means that we explore both cofactors of the circuit with respect to the high activity variable with higher probability. Thus, the suspected inputs need to be provided with high activity and can

TABLE II
Trojan Detection on ISCAS85 Benchmarks

ISCAS Benchmark	Benchmark with Trojan	IP/OP	Fingerprint
c17	1e17	5/2	$\{1,3\}$
c17	2e17	5/2	$\{2\}$
c1355	1e1355	41/32	$\{176\}$
c1355	2e1355	41/32	$\{8,22,64,113,141,183,218,229,230,233\}$
c1908	1e1908	33/25	$\{1,4,22,25,28,31,34,40,46,94\}$
c1908	2e1908	33/25	*Not Detected*
c432	1e432	36/7	$\{4,14,17\}$
c432	2e432	36/7	$\{4,56,66\}$
c499	1e499	41/32	$\{1,9,21,29,37,41,73,85,113,117,131,137\}$
c499	2e499	41/32	$\{1,33,49,73,77,97,109,113,125\}$
c6288	1e6288	32/32	*Not Detected*
c6288	2e6288	32/32	$\{1,52,69,86,103,273,290,307,324,341,358,375\}$

be harmlessly clubbed with inputs which are expected to be benign.

B. Circuit with Multiple Outputs

In case, a circuit has more than one output, a circuit would be considered infected with a trojan if there exists some input assignment such that atleast one output of CUT is different than the corresponding output of the design.

A simple way to achieve this is to decompose the circuit into a set of Boolean function representing *cone of each primary output* and then, applying the above technique for each output separately. We currently use this in our experiments with ISCAS85 benchmarks.

If a fingerprint is detected that distinguishes a single output, we can also treat it as fingerprint of the trojan infection. If after N simulations, the outputs of the CUT and the design do not differ, then we require that the *confidence* of the hypothesis corresponding to each primary output be at least as much as the required confidence of claiming that the CUT is trojan free. Thus, if we require that a CUT be said to be trojan free with confidence 95%, then for each output of the CUT, we should be able to show that the hypothesis that the mean of the output being 1 is equal to the probability signature is accepted with significance 0.05.

C. Extent of Infection and Hardness of Detection

A key property of our technique is that the greater the extent of functional modification of the CUT due to an infection with a trojan, the easier it would be to detect the infection. This is due to the property of our signature. If two Boolean functions differ at more minterms, then their signatures are further apart. Hence, if an infection results in toggle of more number of minterms, then the signature of the infected circuit will be very different from the probability signature of design. Consequently, we will be able to find a fingerprint with higher probability. Also, it is less likely to reach a high confidence level with a seriously infected circuit since the signatures of the infected CUT and design would be far apart.

V. Experiments

We now present experimental evaluation of our technique using ISCAS benchmark circuits. We added extra logic to these circuits randomly to change their functionality and to make them infected circuits. Then, the infected circuit and the original ISCAS benchmark were given as inputs to the trojan detection algorithm presented in this paper. Though in a practical setting, we expect that a hardware simulator would be used to stimulate the circuits with inputs generated from the probability distribution corresponding to the signature, in our experiments we do a software simulation of the circuit.

The results of our experiments are presented in Table II. The first column is the name of the ISCAS circuit and the second is the infected circuit. The third column gives the number of inputs and outputs in the example. The fourth column gives the fingerprint which distinguishes the infected circuit from the original circuit. All inputs mentioned are the inputs assigned 1 in the fingerprint; the remaining inputs are assigned 0. The simulations were run with a timeout of 30 minutes. We were able to detect trojans in 10 out of the 12 examples.

The ISCAS benchmarks and the infected circuits as well as the experimental data are available from first author's website[1].

VI. Conclusion

The technique presented in this paper for trojan detection can be used to verify any black-box implementation against a white-box design. If the analysis is able to find a fingerprint that distinguish the implementation and the design, then this fingerprint can be used for post-analysis like debugging to further localize the cause of difference.

[1] www.eecs.berkeley.edu/~jha/trojan_hase.tar

In context of trojan, fingerprints can be useful for *invasive* techniques like X-ray tomography which can locate the exact difference using a layer-by-layer comparison of the chip and the design.

Though induction and bounded model checking can be used to convert a sequential circuit into a combinational circuit, we are trying to develop a randomized slicing based approach to unroll only parts of circuit to make our technique more scalable on sequential circuits. It would be interesting to decouple the problem of hunting for a trojan in a sequential circuit into two separate problems - testing memory elements and testing the combinational circuit. Another direction of future work is to use the fingerprint to provide more localization information regarding the nature of trojan infection.

Acknowledgment

The first author acknowledges the support of the Berkeley Fellowship for Graduate Studies from UC, Berkeley.

References

[1] S. Adee, "The hunt for the kill switch," *Spectrum, IEEE*, vol. 45, no. 5, pp. 34–39, May 2008.

[2] Website. (June 2003) White paper: National security aspects of the global migration of the us semiconductor industry. [Online]. Available: http://lieberman.senate.gov/documents/whitepapers/semiconductor.pdf

[3] website. DARPA BAA06-40, TRUST for integrated circuits. [Online]. Available: http://www.darpa.mil/BAA/BAA06-40mod1.html

[4] Website. (February 2005) The U.S.D. of defense, defense science board task force on high performance micro-chip supply. [Online]. Available: http://www.acq.osd.mil/dsb/reports/200502HPMS Report Final.pdf

[5] D. Agrawal, S. Baktir, D. Karakoyunlu, P. Rohatgi, and B. Sunar, "Trojan detection using IC fingerprinting," in *SP '07: Proceedings of the 2007 IEEE Symposium on Security and Privacy*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 296–310.

[6] M. Banga, M. Chandrasekar, L. Fang, and M. S. Hsiao, "Guided test generation for isolation and detection of embedded trojans in ICs." in *ACM Great Lakes Symposium on VLSI*, V. Narayanan, Z. Yan, E. Macii, and S. Bhanja, Eds. ACM, 2008, pp. 363–366.

[7] F. Wolff, C. Papachristou, S. Bhunia, and R. S. Chakraborty, "Towards trojan-free trusted ICs: Problem analysis and detection scheme," *Design, Automation and Test in Europe, 2008. DATE '08*, pp. 1362–1365, March 2008.

[8] C. Fagot, O. Gascuel, P. Girard, and C. Landrault, "On calculating efficient lfsr seeds for built-in self test," *Test Workshop 1999. Proceedings. European*, pp. 7–14, 1999.

[9] W.-T. Cheng, M. Sharma, T. Rinderknecht, L. Lai, and C. Hill, "Signature based diagnosis for logic bist," *Test Conference, 2006. ITC '06. IEEE International*, pp. 1–9, Oct. 2006.

[10] E. M. Clarke, A. Biere, R. Raimi, and Y. Zhu, "Bounded model checking using satisfiability solving," *Formal Methods in System Design*, vol. 19, no. 1, pp. 7–34, 2001. [Online]. Available: citeseer.ist.psu.edu/clarke01bounded.html

[11] S. K. Kumar and M. A. Breuer, "Probabilistic aspects of boolean switching functions via a new transform," *J. ACM*, vol. 28, no. 3, pp. 502–520, 1981.

[12] V. D. Agrawal and D. Lee, "Characteristic polynomial method for verification and test of combinational circuits," in *VLSID '96: Proceedings of the 9th International Conference on VLSI Design: VLSI in Mobile Communication*. Washington, DC, USA: IEEE Computer Society, 1996, p. 341.

[13] P. Ashar, A. Ghosh, and S. Devadas, "Boolean satisfiability and equivalence checking using general binary decision diagrams," *Computer Design: VLSI in Computers and Processors, 1991. ICCD '91. Proceedings, 1991 IEEE International Conference on*, pp. 259–264, Oct 1991.

[14] E. Goldberg, M. Prasad, and R. Brayton, "Using sat for combinational equivalence checking," in *DATE '01: Proceedings of the conference on Design, automation and test in Europe*. Piscataway, NJ, USA: IEEE Press, 2001, pp. 114–121.

[15] M. Blum, A. K. Chandra, and M. N. Wegman, "Equivalence of free Boolean graphs can be decided probabilistically in polynomial time," vol. 10, no. 2, pp. 80–82, Mar. 1980.

[16] S.-C. Wu and C.-Y. Wang, "Peach: A novel architecture for probabilistic combinational equivalence checking," *Very Large Scale Integration, 2006 IFIP International Conference on*, pp. 104–109, Oct. 2006.

[17] J. Jain, J. A. Abraham, J. R. Bitner, and D. S. Fussell, "Probabilistic verification of boolean functions," *Formal Methods in System Design*, no. 1, pp. 61–115, 1992.

[18] D. C. Montgomery and G. C. Runger, *Applied Statistics and Probability for Engineers*. Wiley Text Books; 3rd edition, ISBN: 0471204544, August 2002.