Proposed Trojan Detection Methods


Use noise modeling to construct a set of fingerprints for an IC family utilizing side-channel information such as power, temperature, EM profiles

Fingerprints are developed using a few ICs, that are later destructively verified

The chips-under-test (CUTs) are verified using statistical tests against the fingerprints

They show Trojans 3-4 orders of magnitude smaller than the CUT can be detected using signal processing techniques

Problem: The problem of Trojan detection essentially reduces to detecting a Trojan signal hiding in the IC process noise, i.e., the small, random, physical and side-channel differences among different ICs produced from the same process.
Proposed Trojan Detection Methods (Agrawal et al)

The identified several challenges:

- To determine a **small and non-redundant** set of tests that provide sufficient coverage of the IC’s functionality
- To determine test patterns that are **comprehensive** and **practical**, and which are capable of distinguishing most Trojans from genuine ICs
- [Destructive verification uses] demasking, delayering and layer-by-layer comparison of X-ray scans with the original mask -- expensive but done on only a few ICs

Experiments: Determine effectiveness of fingerprinting methodology for detecting Trojans by using **power simulations**

- Experimental design: cryptographic circuits implementing the Advanced Encryption Standard (AES) and RSA algorithm
- Trojans investigated: Trojans triggered by timing/clock counting and Trojans triggered by a synchronous/asynchronous comparator
- Trojan sizes: range from 10% to 0.01% of the total IC size
- Noise modeling: noise introduced by process variations (+/- 2%, 5%, 7.5%)
Proposed Trojan Detection Methods (Agrawal et al)

Power consumption:

\[ P = \left( \frac{1}{2} \cdot C \cdot V_{DD}^2 + Q_{se} \cdot V_{DD} \right) \cdot f \cdot N + I_{\text{leak}} \cdot V_{DD} \]

N: switching activity

I_{\text{leak}} \text{ depends only on the number of gates (not switching activity)}

Dynamic power is linearly dependent on the clock frequency and switching activity

Trojan detection by clock speed manipulation: fast vs slow frequency

Figure 1. Genuine (green/grey) and Trojan (blue/black) AES signals at 100MHz (left) and 500KHz (right).

Figure 3. Genuine RSA signal (top: green or grey), process noise(middle: red or dark grey) and Trojan contribution (bottom: black).

Figure 4. Trojan (black) vs. process noise (green or grey) in between two modular multiplications.
Proposed Trojan Detection Methods (Agrawal et al)

What about hiding a Trojan in the signal measurement noise?

They claim measurement noise can be *eliminated* by averaging.

Therefore, they claim the problem degenerates to a **signal characterization** problem. The objective is to characterize the **process noise** and check if the signal for the chip-under-test (CUT) differs from the process noise.

Authors propose the use of **subspace projection** which projects process noise signals from genuine ICs to a subspace where signals from Trojans and genuine ICs differ.
Proposed Trojan Detection Methods (Agrawal et al)

Challenge is guessing how the Trojan may change the genuine signal, otherwise **full characterization** of the process noise is necessary.

Authors propose *advanced signal processing techniques* (Karhunen-Loeve expansion) to find a signal subspace in which process noise is **absent**. Unless the Trojan signals completely ‘live’ in this subspace (unlikely), projecting the Trojan signal to this subspace reveals its presence.

Authors perform experiments on RSA implementations with 3 different sized Trojans.
Proposed Trojan Detection Methods


The main deficiency with parametric testing approaches is sensitivity

- Scaling increases manufacturing process variations
- Larger number of components on a chip decreases the relative magnitude of the electrical signature of each component

The challenge of implementing an effective parametric Trojan-detection method is

- To design it with enough sensitivity to detect small anomalies introduced by Trojans
- Building in a mechanism to filter out the natural electrical variations that occur because of manufacturing process variations
Proposed Trojan Detection Methods (Aarestad, et al)

Contributions:
- Proposed approach is to measure \( I_{DDQ} \) (steady-state current) at multiple places simultaneously across the 2-D surface of the chip.
  
  A region-based \( I_{DDQ} \) method directly addresses the adverse impact of increasing levels of process variations and leakage currents.
  
- Proposed approach uses signal calibration techniques to attenuate and remove PE (process and environmental) signal variation effects.

Experiment:
- A set of chips fabricated in IBM’s 65 nm, 10 metal layer SOI technology are used in the experiments.
- The chips incorporate an array of cells that allow a Trojan to be emulated in one of 4,000 distinct locations on the chip.

The test structure permits control over:
- The position and magnitude of the Trojan current.
- The magnitude and distributional characteristics of the chip-wide leakage current.
Proposed Trojan Detection Methods (Aarestad, et al)

2 TC subset of the 80x50 array

- PWR supply
- 0.9V
- Trojan emulation wire
- Trojan source

- FF1
- FF2
- FF3
- Inverter
- Trojan emulation transistor

- I_T
- I_leak
Proposed Trojan Detection Methods (Aarestad, et al)

Trojan-free leakage current distribution, and emulated Trojan placement, labeled 1 through 9 in the figure.

Scan chain allows the off state of the shorting inverters to be configured into a high leakage (HL) or low leakage (LL) state.
Proposed Trojan Detection Methods (Aarestad, et al)

'Golden model' is defined by the actual chips (not simulation experiments) by disabling all emulated Trojans

Four branch currents through PP₀ through PP₁₁ (and global currents) are measured for each chip

Emulated Trojan experiments enable one Trojan emulation transistor

TESM voltage is swept from 0.8 V to 0.89 V in 10 mV steps (10 steps)
For each step, 4 branch currents (and global current) measured
Trojan current varied from 8 uA to 62 uA

All together, each chip produces 91 data sets, 1 Trojan-free data set and 90 emulated Trojan data sets (9 Trojans * 10 TESM voltages)

With 45 chips, there are a total of 45 Trojan-free data sets and 4,050 emulated Trojan data sets.
Proposed Trojan Detection Methods (Aarestad, et al)

Our statistical analysis is implemented using scatterplots, where one PP current is plotted against another.

Regression involves deriving a ’best fit’ line through the Trojan-free data points. 3 sigma statistical limits (parabolic curves) can then be derived.

A Trojan is detected if it’s data point falls outside the limits in at least one of the six scatterplots.
Proposed Trojan Detection Methods (Aarestad, et al)

Calibration

Dispersion in Trojan-free data points caused by
- chip-to-chip variations in the power grid resistance
- series resistance variations from PPs to external power supply

Special **calibration circuits** (CCs) are inserted into the design
- They are identical to those shown earlier but without the Trojan emulation transistor and wire
- They are inserted under each of the PPs

Calibration data is collected by
- Enabling each of the CCs (one at a time) and measuring the 4 branch and global currents
- A matrix of calibration currents is constructed from *normalized* branch currents, where each is divided by the corresponding global current

This matrix (one for each chip) is used to calibrate data collected under the emulated Trojan tests
Proposed Trojan Detection Methods (Aarestad, et al)

Calibration matrix and calibration operation

\[
X = C_x^{-1} \times S
\]

\[
\begin{bmatrix}
    x_{00} & x_{01} & x_{02} & x_{03} \\
    x_{10} & x_{11} & x_{12} & x_{13} \\
    x_{20} & x_{21} & x_{22} & x_{23} \\
    x_{30} & x_{31} & x_{32} & x_{33}
\end{bmatrix}
= \text{inv} \begin{bmatrix}
    a_{00} & a_{01} & a_{02} & a_{03} \\
    a_{10} & a_{11} & a_{12} & a_{13} \\
    a_{20} & a_{21} & a_{22} & a_{23} \\
    a_{30} & a_{31} & a_{32} & a_{33}
\end{bmatrix} \times \begin{bmatrix}
    r_{00} & r_{01} & r_{02} & r_{03} \\
    r_{10} & r_{11} & r_{12} & r_{13} \\
    r_{20} & r_{21} & r_{22} & r_{23} \\
    r_{30} & r_{31} & r_{32} & r_{33}
\end{bmatrix}
\]

Transformation matrix

Chip data

Data collected from 'golden' simulation model

Data from chip using Trojan test

'Corrected' data
Proposed Trojan Detection Methods (Aarestad, et al)

Regional leakage current variations decreases Trojan detection sensitivity

Chip C₁

Chip C₂
Proposed Trojan Detection Methods (Aarestad, et al)

Regression Analysis for Trojan detection:

450 points per scatter plot (45 chips times 10 TESMs)

Trojan #1

Trojan #2

More Trojans detected

Trojan #3

Trojans detected

Trojan #4

Uncalibrated Calibrated

Trojan #5

Uncalibrated Calibrated

Trojan #6

Uncalibrated Calibrated

Trojan #7

Uncalibrated Calibrated

Trojan #8

Uncalibrated Calibrated

Trojan #9

Uncalibrated Calibrated

450 pointsscatter plot (45chipstimes10TESMs)
Proposed Trojan Detection Methods (Aarestad, et al)

Before and after calibration:

Uncalibrated

Calibrated

Regression: Uncalibrated

Regression: Calibrated

Number of Chips

Trojan Current (µA)

Trojan #
Proposed Trojan Detection Methods


Authors identify three possible triggering mechanisms:
- Rare value triggered
- Time-triggered
- Both

Two components:
- Triggering: occurs only under rare conditions
- Payload activation logic

Insertion is likely to nodes with low controllability and observability

The adversary disables the Trojan when the test enable signal is driven
Therefore, scan-based designs do NOT help improve security and functional test must be used.
Proposed Trojan Detection Methods (Wolff et al)

They define a *trojan test vector* as a **trigger** vector that propagates the payload to the circuit output.

A **trigger vector** triggers the Trojan only.

![Diagram](image1)

(a) AND gate, Test Set = \{01, 10, 11\}

![Diagram](image2)

(b) Hacked Circuit, Passes s-a-f test

Figure 2. Trojan Circuit evading Single Stuck Fault testing

Figure 3. The Time Bomb
Proposed Trojan Detection Methods (Wolff et al)

They define the nodes targeted by their technique using 2 rules:

• The target nodes are all combinations of $q$ nodes that attain a specific logic value with frequency $\leq f_{th}$, where $q$ is the number of Trojan inputs and $f_{th}$ is the probability that those nodes are toggled.

• Insert payload (gates that change functionality) on nodes that have low probability of propagating to an circuit output.

They use logic and fault simulators to identify a set of target nodes and payload nodes, and then use ATPG to determine the trigger test vectors. Details of the ATPG strategy are not provided.

They admit their strategy can be effective in detecting most small combinational Trojans.
Proposed Trojan Detection Methods


The authors analyze the amount of time it takes to 1) generate a transition in a functional Trojan, partially active it with test vectors and 2) trigger a hardware Trojan.

They propose a dummy FF insertion process to increase Trojan activity and ultimately reduce Trojan activation time.

Trojan inputs are likely connected to nodes with low controllability and/or observability.

A Trojan cone is used to describe the logic gates driving the inputs to a Trojan gate.

![Figure 1: Two Trojan cone examples: (a) Trojan 1 and (b) Trojan 2](image)
Proposed Trojan Detection Methods (Salmani et al)

Application of random patterns show that different numbers of transitions occur in the Trojan gate, that largely depend on Trojan cone configuration.

Probability analysis can determine the likelihood of a Trojan gate output switching.

![Diagram](image1.png)

\[ \text{Transition probability} P_{tr} = \frac{138}{256} \times \frac{118}{256} = 0.25 \]

\[ \text{output} \_\text{prob}_1 = \text{input} \_\text{prob}_1 \times \text{input} \_\text{prob}_2 \]

\[ \text{output} \_\text{prob}_0 = (1 - \text{output} \_\text{prob}_1) \]

They use a geometric distribution function to compute the average number of clock cycles it takes to generate a transition in the Trojan gate \((P^{-1} - 1)\).

Large differences in the output probabilities reduces the transition probability significantly, therefore, it is best to try to balance these.
Proposed Trojan Detection Methods (Salmani et al)

The authors propose to insert *dummy* FFs to maintain a balance. This eliminates *hard-to-activate* sites, which in turn, increases the probability of switching (full or partial activation) in the Trojan.

So, this eliminates the need to focus on *rare* conditions, as in Wolff et al.

A threshold probability, $P_{TH}$, is defined to select nets for dummy FF modification. The choice trades-off *area overhead* versus *Trojan transition generation time*.

Also, when transient current methods are used to detect the Trojan, then *partial activation* is sufficient, and the larger the number of partial activations, the better.

The authors give an expression that trades off test time, area overhead and the number of Trojan transitions.
Proposed Trojan Detection Methods


The path delays of nominal chips are collected to construct a series of fingerprints, that chips are validated against.

They depend on using a sample of chips, apply tests and then destructively validate them.

They carry out simulation experiments on DES IP core in which they introduce 4 Trojans, three are comparators and one a counter Trojan.

The Trojans occupy 0.13% and 0.76% of the total circuit area, respectively.

They also introduce delay variations of up to 7.5% and synthesize the DES circuits without the Trojans (Trojans are added to the netlist afterwards).

Synopsys is used to generate 990 genuine models and 800 Trojan models.
Proposed Trojan Detection Methods (Yier et al)

Synopsys TetraMAX ATPG tool is used to generate 163 patterns, designed to cover as many parts of the chip as possible.

The DES core has 64 outputs and therefore, a total of 10,432 path delays are determined from simulations for each of the models.

The high dimensionality of the data is reduced using principle component analysis (PCA) to determine the major trends in the original data set.

The first three components are selected for analysis.

A convex hull algorithm is applied to the path delays of the genuine models to define the Trojan-free space.

64 convex hulls are generated with each reflecting one aspect of the whole fingerprint of a genuine chip.

Fig. 4: Delay Comparison of Experiment 1
Fig. 5: Delay Comparison of Experiment 2
Fig. 6: Delay Comparison of Experiment 4
Proposed Trojan Detection Methods


In their first paper (HOST 2008), they propose the insertion of shadow registers that are controlled by a phase-shifted version of the on-chip clock.

![Diagram of proposed Trojan detection method]

XOR acts as a comparator and the LOCK block latches a ’1’ when the main register and shadow register differ (which can be read out using scan-chains).
Proposed Trojan Detection Methods (Rai et al)

With knowledge of the clock skew value used when $LOCK$ is set to ‘1’, the combinational delay can be computed for the path-under-test.

The authors focus on analyzing their technique in the presence of significant levels of process variations.

They conduct simulation experiments on a Braun Multiplier using an **two-inverter** chains as a Trojan.

![Diagram showing the section of Braun Multiplier design showing the HTH.](image)

**Trojan**

Figure 3. Section of Braun Multiplier design showing the HTH. The highlighted region marks the place where two-long inverter chain was inserted. DFFPOSX1 is positive edge triggered D-flip flop. HTH is inserted at the input of DFF holding bit 15 of product.

![Graph showing the impact of HTH on bit 15 of output.](image)

**Trojan increases delay**

Figure 4. Impact of HTH on bit 15 of output.
Proposed Trojan Detection Methods (Rai et al)

The *skew-step resolution* is investigated and it was decided that 0.05 ns (50 ps) is needed to detect the insertion of a single inverter.

They do not address test vector generation but decide that *shadow* registers are needed at all outputs.

For each vector, the smallest *skip step* is determined for each shadow register using a simulation model with no Trojans.

The authors introduce both *inter-die* and *intra-die* variations in $V_{th}$ (+-20%) and channel length ($L_{eff}$) in two sets of simulations.

![Graph](image1.png)  
*Figure 7. Bit 5 delay distributions with $V_{th}$ variations and HTH at [3,2].*

![Graph](image2.png)  
*Figure 8. Bit 9 delay distributions with $V_{th}$ variations and HTH at [3,2].*
**Proposed Trojan Detection Methods**


The authors propose a circuit partition based approach to detect and locate embedded Trojans. They also propose a power profile based method for refining the candidate regions that may contain the Trojan.

They define a region as a *structurally connected* set of gates.

They compute the total power profile of a genuine circuit,

\[ P = CV^2f \]

Their approach consists of two major steps:

- *Region-based Partition*: Determine appropriate regions for analysis.
- *Relative Toggle Count Magnification*: Generate a suitable input vector set that maximizes the partial relative power consumed in each region.
Proposed Trojan Detection Methods (Banga et al)

A circuit with 5 regions

The region surrounding a gate comprises all the transitive fanin and fanout gates that are within the defined radius

Once the regions are selected, ATPG is used to create an activity peak in each region, while minimizing switching activity in the rest of the IC

They acknowledge that detection is possible only if the difference in activity in Trojan and genuine chips is larger than process variation
Proposed Trojan Detection Methods (Banga et al)

Regions of larger differences

Blue: random vectors
Brown: author’s vectors

Fig. 9. TCM(Radius 3, flip-flop Count 3) for s3271

Fig. 10. TCM(Radius 4, flip-flop Count 5) for s3271

(Graphs have no annotation in paper: x-axis are vector groups, y-axis is percentage change)
**Proposed Trojan Detection Methods**


The authors propose a *randomization* based method to probabilistic compare the functionality of the implemented circuit with the original design.

To determine if a manufactured chip conforms to its design (or contains a Trojan) by *functionally activating* the Trojan.

They find a probability distribution on the *inputs* such that the probability distribution of the *output* is *unique* for every functionally distinct circuit.

Hypothesis tests is used to statistically infer the presence of a Trojan.

The result is either an input pattern that distinguishes a Trojan circuit from the design or a confidence level that no Trojan exists.

They define a *characteristic polynomial* of a circuit and prove that two Boolean functions $f$ and $g$ are equal *if and only if* their char. poly. are identical.