A Review and Implementation Proposal of Self-Referenced-based Hardware Trojan Detection

Alex Yeoh

Wright State University

Dayton, Ohio, USA

Yeoh.3@wright.edu

**Abstract – This review and implementation proposal provides a thorough summary of “*Hardware Security and VLSI Design Optimization*” chapter 4 by Dr. Xue. It then provides some recommendations on how to improve the chapter before proposing a simple implementation of Dr. Xue’s Hardware Trojan Detection Method.**

I. INTRODUCTION

The topic of hardware security is important to everyone because microelectronics are everywhere; from cameras to cars and everything in-between. With how ubiquitous microelectronics are, it is important for companies to find hardware trojans to ensure their products at least work properly and, if their product contains private information, also prevent any information theft. To prevent these hardware trojans, it is vital to first detect them with hardware trojan detection methods like the one developed by Dr. Xue in chapter 4 of their dissertation “*Hardware Security and VLSI Design Optimization*” [1].

II. SUMMARY

*A. Section 4.1 – Introduction [1]*

This section describes how various other power analysis-based *hardware* *trojan* (HT) detection works and their drawbacks. The drawbacks described include the need for the HT to be relatively large to be detected and the need of a perfect circuit which is unreliable because of manufacturing uncertainties. It finally describes the process of power analysis-based HT detection in this chapter of Dr. Xue’s dissertation as solving a system of equations that consist of the summation of the products of a manufacturing variability factor and ideal static power for a gate with a HT variable.

*B. Section 4.2 – Background Theory and Accounting for Process Variation [1]*

A *circuit* *under* *test* (CUT) is separated into multiple different segments with every segment having separate power rails so segments can be individually examined, controllable primary inputs, and with an input from the on-chip clock tree in segment 0. Each segment without a trojan is described with equation 1 below and each segment with a trojan is described with equation 2 below.

|  |  |
| --- | --- |
| Symbols | Meaning |
| *Q* | Segment Number |
| *IN* | Input Pattern |
| *T* | Operating Time |
| *i* | Component Number |
| *k* | Total Component Number |
| *Pmes* | Power Measured |
| *p* | Static Power |
| *pi* | Ideal Static Power of Component *i* |
| *nv* | Operation Variation on Static Power |
| *nvi* | Operation Variation on Static Power of Component *i* |
| *ns* | Signal Noise |
| *nsi* | Signal Noise of Component *i* |
| *pT* | Extra Power consumed by the Trojan |

Where every component of equations 1 and 2 are explained in table 1 below.

Table 1: Table Explaining Symbols used in equations 1 and 2.

Equations 1 and 2 can be simplified in this use case. Signal noise can be ignored due to the averaging of many power measurements on the same *integrated* *circuit* (IC). Manufacturing uncertainties cause *process*, *voltage*, and *temperature* (PVT) variation. But after manufacturing, the PVT variation for any given IC remains constant, which allows the operation variation (λ) on static power to become a constant with the symbol. With these changes, equations 1 and 2 simplify to equations 3 and 4 below.

Because during the trojan detecting process, we don’t know if the CUT has a trojan, equation 3 is used. Due to ideal static power of a component being constant, if the CUT segment has a trojan, the extra power consumed by the trojan manifests itself as a larger λ. The author then acknowledges the pros and cons of their HT detection process which can be seen in table 2 below.

|  |  |
| --- | --- |
| Pros | Cons |
| HT detection method can be applied with zero-overhead. | Requires many I/O pins |
| Genuine chip is not needed. |  |
| Location of HT can be estimated. |  |
| Partitioned CUT increases detection accuracy. |  |
| Detection accuracy is adjustable by size of partitioned segments. |  |
| Computational complexity is optimizable based on required detection accuracy. |  |
| Designer can adjust design to better check vulnerable areas of their design. |  |

Table 2: Table Listing Dr. Xue’s Listed Pros and Cons for their HT Detection Process.

This is then followed by a further explanation on why this HT detection process targets static power and how it works. Static power is the value used for HT detection because even if a HT is dormant, it still contributes to static power of an IC and PVT variations have a much stronger impact on dynamic power compared to static power. The critical point in how the HT detection works is due to the Gaussian distribution of static power due to PVT variations, if 3σ is used as a cutoff, only 0.26% of parts which don’t have a HT will be mistakenly rejected as having a HT. To demonstrate that input patterns do not affect static power while PVT does, Dr. Xue performed an experiment graphing the static power of a 1-bit full adder simulated with 3 different input patterns and many different PVT variations resulting in figure 1 below.

A diagram of a graph

Description automatically generated

Figure 1: Dr. Xue’s Graph Generated for Static Power with Different Input Patterns and PVT Variations.

The red line in the middle represents the points where the ratio of measured to simulated static power is the same for all three input patterns. This means input patterns do not affect static power while PVT variations do because if input patterns inputs affected static power, the points would be spread further apart from the red line. While if PVT variations did not impact static power, all the points would group together into bubbles for each input pattern. This section ends with Dr. Xue reiterating λ*i* is in equation 3 and 4 because of PVT variations impact on static power and mentions that the clock-tree has a similar equation where static power for the clock-tree is λ*i*p*clk*.

*C. Section 4.3 – Implementation of HT Detection Algorithm, Optimization, and Necessary Overhead [1]*

This section begins with figure 2 below which illustrates the algorithm used for HT detection.

A diagram of a computer program

Description automatically generated

Figure 2: Algorithm Flow Chart for HT Detection.

Step 1 is requiring the CUT to be in a form that can be analyzed with the discussed HT detection process. Step 2 then requires the ideal static power for each gate with the desired input patterns to be generated. Step 3 is measuring the static power for each segment with the clock tree separately. Step 4 calculating the λi and λclk in equation 5 below by solving the system of equations generated with the data from step 3 by minimizing the sum squared difference seen in equation 6 while limiting λ*i* to between 0.7 and 1.3.

Where j is the static state, m is the total number of static states, is the static power of the CUT with the static state j and the remaining symbols retain their previous meaning from table 1. Step 5 simply says to repeat steps 3 and 4 for every segment in the CUT. Step 6 is a check to see if the clock λ for any given segment is inside the error bounds determined by equation 7 below.

If the segment is within the bounds generated by equation 7, the segment is considered trojan-free and is considered trojan-attacked otherwise. The dissertation follows this section with an example; because this is a summary, it would not make sense to attempt to summarize this portion and it is advised to read the example to follow along with the dissertation with everything explained as verbosely as given. The example section does, however, contain the statement, “PVT variation affects circuit static power consumption within 30% - (𝜆𝑖) should be constrained to 0.7 < 𝜆𝑖 < 1.3”, and figure 3 below which merits discussion in a summary.

A graph with lines and numbers

Description automatically generated

Figure 3: Graph of # Equations vs Deviation from Optimal Clock Operational Variation

As seen in figure 3, there is a general tread downwards as the number of equations used increases where the maximum is limited by the total number of feasible static states. This shows that as the number of equations increases, the HT detection becomes more accurate in marking all trojan-attacked segments as the min clock λ becomes smaller in equation 7. Dr. Xue then performs an experiment with various benchmark circuits to determine the false negative rate in relation to Λq as seen below in figure 4.

A graph of a number of objects

Description automatically generated with medium confidence

Figure 4: Graph of Λq Threshold vs False Negative Rate.

As seen above in figure 3, although a smaller Λq would increase accuracy and lower the false positive rate, a Λq that is too small would not be useful as it would create an issue with false negative rates as seen in figure 4. As the actual false positive and false negative rates would vary depending on various conditions like the size difference between the trojan and host-circuit, and design of the host-circuit and the design of the HT, there would not be a perfect Λq to use for all circuits, figure 4 seems to imply a threshold percentage of around 10% would likely work well for many circuits as it reached a 0% false negative rate for the set of benchmark circuits Dr. Xue tested with. A summary of when a segment can be asserted to be trojan-attacked can be seen in a snippet of Dr. Xue’s dissertation below in figure 5.

A white background with black text

Description automatically generated

Figure 5: Criteria for Asserting Trojan-Attacked based on Dr. Xue’s Dissertation.

The algorithm mentioned above in figure 2 can be abstracted into the pseudo-code from Dr. Xue’s dissertation in figure 6 below.

A paper with text and symbols

Description automatically generated

Figure 6: Psuedo-code from Dr. Xue’s Dissertation for Implementing the HT Detection Algorithm

Section 4.3.2 from Dr. Xue’s dissertation is an entire section that follows an example and will not be summarized for the same reason mentioned shortly prior to figure 3. The next section of the dissertation describes a process that can be used to simplify the calculations done with equation 6. The simplification involves grouping multiple gates together and considering them as a single gate which would reduce the number of λ needed in equation 6. Although grouping gates together would simplify the calculations, it would also reduce the accuracy of detecting HTs as seen below in the analysis seen in figure 8 below of CUT2 in figure 7 below.

A diagram of a block tree

Description automatically generated

Figure 7: CUT2

A table with numbers and a number on it

Description automatically generated

Figure 8: Analysis of CUT2 with Segment 1 with and without HT and Detectable HT Size Relative to CUT2

As seen above in Figure 8, as the group size increases, the side of the HT would also need to increase to be detected. To Further illustrate this point, figure 9 below presents some similar information in the form of a graph.

A graph of a group size

Description automatically generated

Figure 9: Group Size vs # Variables and Group Size vs Detectable HT Size

Dr. Xue describes the necessary overhead to implement his proposed HT detection method. Due to the need of stable nominal supply voltages, many circuits already have many I/O pads for all the power rails. If the CUT has separate I/O pads for all the power segments, there is no overhead to implement this HT detection method. If the CUT has more segments than the I/O pads can account for, extra control circuits are required to split existing power rails into more segments but will not impact the efficacy of the HT detection method.

III. ANALYSIS

The detection method designed by Dr. Xue is comparable to other HT detection methods as seen below in figure 10 which is from table 4.6 in Dr. Xue’s dissertation.

A table with numbers and text

Description automatically generated

Figure 10: HT Detection Comparison

The chapter itself is quite dense and difficult to read, but the solution is very logical and understandable. It simply states that all non-floating components add to the static power, so detecting a HT can be done by determining a range of accepted static power based on simulated PVT variation and statistical analysis and everything outside the range can be considered trojan-attacked. The only downside to this HT detection method is that if the segment tested is too large, the trojan can go undetected; but the solution proposes segmenting the CUT to minimize this issue and this issue is not unique to this HT detection method. To improve on how difficult the chapter is to read, short one-line explainers could be added for terms like typical-typical corner analysis and Monte Carlo analysis.

IV. PROPOSAL

*A. Objective*

The objective is to be able to implement the proposed HT detection method.

*B. Methodology*

A simple circuit will be built following figure 11 below and multiple tests can be performed where a HT is added to a segment.

A diagram of a tree diagram

Description automatically generated

Figure 11: Simple Circuit for HT detection implementation

Tools:

Cadence Virtuoso

Proprietary Data for Monte Carlo analysis / A program to generate fake data to simulate Monte Carlo analysis.

MATLAB

Steps:

1) Build the circuit from figure 11

2) Perform typical-typical corner analysis

3) Add a HT to one of the segments

If proprietary data for Monte Carlo analysis is available,

4) Perform Monte Carlo analysis

If no proprietary data for Monte Carlo analysis is available,

4.1) Generate randomized data with the values using the Java program depicted in figure 12 below

|  |
| --- |
| import java.io.File;  import java.io.FileWriter;  import java.util.ArrayList;  import java.util.Random;  public class FakeMonteCarlo {  // Objects used repetedly  static Random rng = new Random();  static String newLine = "\n";  static String fileExtension = ".txt";  // User modifies main to generate randomized parameters for  public static void main(String[] args) throws Exception {  // Input parameters for each fake component needed  int dataPoints = 10;  int decimalPoints = 2;  generateData(0, 5, 5, dataPoints, decimalPoints); // First transistor's random values  generateData(1, 5, 5, dataPoints, decimalPoints); // Second transistor's random values  }  // Generates fake data points and outputs it to a file  public static void generateData(int fileNum, double mean, double distFromMean, int dataPoints, int decimalPoints)  throws Exception {  ArrayList<Double> data = new ArrayList<>();  // Generate random data points  for (int i = 0; i < dataPoints; i++) {  data.add(rng.nextDouble((mean - distFromMean), (mean + distFromMean)));  }  // Output to file  FileWriter writeTo = new FileWriter(new File(fileNum + fileExtension));  for (Double point : data) {  writeTo.append(String.valueOf(Math.floor(point \* 10 \* decimalPoints) / (10 \* decimalPoints)));  writeTo.append(newLine);  }  writeTo.close();  }  } |

Figure 11: Example code that can generate random value for a simulation of Monte Carlo analysis.

4.2) Input the generated random values for as widths for the transistors to simulate PVT variation

5) Use MATLAB to solve for λ

6) Determine if / where the CUT is trojan-attacked following figure 5 above

*C. Expected Outcome*

The segments without the HT do not fail the requirements from figure 5, and the segment with the HT fails the requirements from figure 5.

**References**

[1] H. Xue, “Hardware Security and VLSI Design Optimization Hardware Security and VLSI Design Optimization,” Ph.D. dissertation, Dept. Elect., Wright State Univ., Dayton, OH, USA, 2018. Accessed: Feb. 19, 2024. [Online]. Available: https://corescholar.libraries.wright.edu/cgi/viewcontent.cgi?article=3347&context=etd\_all