FPGA Based Verification for the Difficulty of Delay Based Hardware Trojan Detection

Fatma Nur Esirci[1], Hasan Mutlu[1], and Alp Arslan Bayrakci[1]

[1]Computer Engineering Department, Gebze Technical University, Turkey
fesirci, h.mutlu2018, abayrakci@gtu.edu.tr

Abstract

Hardware Trojan insertion during the manufacturing phase is a crucial problem for chip designers. There are several approaches to detect the Trojan in circuits without destructing the chip structure. One of them is delay based hardware Trojan detection. However, this technique suffers deeply from the variations and it is a question whether the Trojans can be detected as the variations hide the effect of the Trojan. This paper, rather than simulation tools, performs experiments on actual FPGA chips to demonstrate and verify the contribution of both inter and intra die variations as well as the effect of hardware Trojan on delay. Lastly, the ability of delay based Trojan detection backed up by the ring oscillator is investigated with different sizes of the Trojan.

1. Introduction

Due to the economical reasons, most of the manufacturing is handled by far-east countries and it is very hard to perform all manufacturing domestically even for the most developed countries. As a result, hardware Trojan (HT) insertion by an adversary during the integrated circuit manufacturing phase is a crucial problem. The inserted Trojans can easily cause circuit failure or change its functionality, which may be disastrous especially for the security critical designs.

There are lots of different approaches that deal with hardware Trojan detection problem [1, 2], but most of the literature rely on simulations mostly based on Electronic Design Automation (EDA) tools from companies such as Cadence and Synopsys.

The researchers also work up the power based HT detection methods [3], which try to detect Trojan existence by investigating its trace on supply current whereas delay based detection methods try to reveal delay differences due to the Trojan insertion. The main issue of the delay based HT detection is the effect of the variations that can hide the effect of HT.

There are different delay based detection methods in the literature to overcome this problem [4–9]. For that purpose they apply different variation models some of which are loose and show their detection rates according to the simulation results. However, the results are as accurate as the underlying variation models and utilized tools. It is claimed in [10] that assuming a realistic variation model it is too difficult to dect the Trojans solely looking at path delays. In order to verify this difficulty, our goal is to detect the actual effect of variations as well as the effect of the inserted Trojan on delay utilizing actual manufactured chips. Therefore, we carry out tests on Intel FPGA Cyclone-V chips and reveal the difficulty of delay based Trojan detection. Moreover, we experimentally show that simple variation capturing methods like ring oscillator proposed by [3] for power based Trojan detection, does not avail delay based Trojan detection on actual FPGA chips unless the Trojan is too large.

Section 2 starts with the problem formulation, Section 3 explains the circuitry and measurement method to be used for the experiments, Section 4 reveals the actual delay variations according to the FPGA delay measurements, Section 5 shows the effect of the Trojan on delay, Section 6 demonstrates the results when a ring oscillator is used to suppress variations, Section 7 clarifies the effect of Trojan size on detection performance and lastly Section 8 summarizes the results.

2. Problem Formulation

A path in a digital circuit is composed of gates and its delay is measured as the sum of individual delays of the gates included in that path. However, if an adversary inserts a hardware Trojan into a path, the new path delay has to contain the delay of the payload part of the Trojan as it is along the attacked path. Fig. 1 demonstrates such a Trojan-inserted path [10].

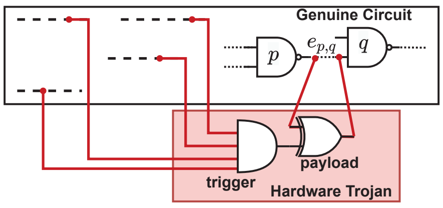

Fig. 1. Hardware Trojan Model.

The payload part is inserted into the path and directly affects the delay. The trigger part is for the Trojan to stay in passive state so that the payload does not change the functional output of the circuit and thus, it cannot be detected by conventional logic tests. It inverts the path signal only whenever the output of the trigger gate becomes logic-1. However, delay based methods are advantageous in the sense that they can detect the Trojan even in the passive state. Because the payload part is already inside the path. As a result, the delay of the Trojan inserted path becomes:

$$d_{\hat{P}}(X) = \sum_{i=1}^{k} d_{n_i}(X) + d_{n_{load}}(X) \tag{1}$$

The aim of delay based hardware Trojan detection is to detect that additional component ($d_{n_{load}}(X)$) belonging to the Trojan.

In (1), X is a vector that denotes the circuit parameters like gate length and threshold voltage. These parameters change from chip to chip (inter die) and even in the same chip (intra die). The cause of that change is the non-negligible and unavoidable manufacturing process variations. As a result, the delay of a path may differ, which results in delay variations. These delay variations can be so high that they can hide the additional delay of the inserted hardware Trojan, i.e. $d_{n_{load}}(X)$.

It is very difficult to model process variations. The models are either loose or demanding too much processing power to get the resultant effect on circuit delay. In this paper, instead of using variation delay models and simulation tools, in order to see the actual effect of the process variations and the hardware Trojan, we measure the delays of the paths on actual FPGA chips. Before presenting the results, the circuitry used for measuring the path delays should be explained.

3. Path Delay Measurement Circuitry

In order to compute the delay of a path inside an FPGA, a simple sequential circuit is designed with the state machine shown in Fig. 2. This circuit is connected to the input and output of the path, delay of which is desired, i.e. the path under test. All the side inputs of the gates on the path under test are initially set to non-controlling values[1], therefore, each input change has to propagate until the end of the path, i.e. until the output.

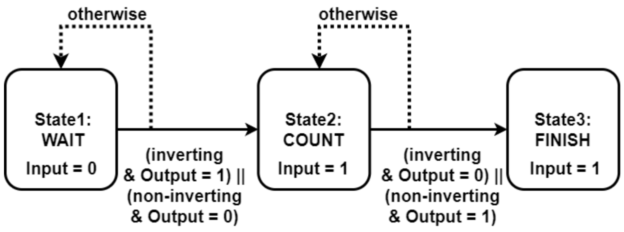
otherwise
otherwise
State1: WAIT
Input = 0
(inverting & Output = 1) || (non-inverting & Output = 0)
State2: COUNT
Input = 1
(inverting & Output = 0) || (non-inverting & Output = 1)
State3: FINISH
Input = 1

Fig. 2. Simple State Machine.

As Fig. 2 demonstrates, in the first state, we apply a logic-0 to the input of the path under test and wait until the output gets stabilized. If it is a non-inverting path it stabilizes at logic-0, else if it is an inverting path it stabilizes at logic-1. Then, in the second state we apply logic-1 to the input and meanwhile we start a counter. The last state waits for the resultant change at the output and stops the counter when that occurs. The value at which the counter is stopped provides us the number of cycles in between the input change and the corresponding output change. If we multiply that value with the clock period, we get the delay of the path under test.

The clock speed is taken to be the maximum speed allowed by the FPGA under test for best resolution. Assuming that max permitted clock speed is represented by f_{max} and the final value of the counter is N, the resultant path delay becomes N/f_{max}. The minimum delay difference that can be computed by such a setup is $1/f_{max}$, which puts a lower limit for the size of the path whose delay is measurable.

As an example path, we use a path from c7552 that belongs to well-known ISCAS'85 benchmark circuits [11]. The path has 35 logic gates including NAND, NOR, AND, OR gates and NOTs. Actually, the types of the gates can not affect the result for FPGA as all gate types are converted to LUTs inside FPGA.

There is a problem in putting that path inside FPGA with connecting all side inputs to non-controlling values, because the optimization tools of FPGA software, i.e. Quartus, automatically applies optimization and as a result, converts the path into a single buffer for non-inverting paths and into an inverter for the inverting paths. To avoid all the optimization steps, `/*synthesis keep*/` must be inserted as a comment line for each row in the Verilog where an input or an output is defined.

We utilize Intel Cyclone-V 5CEBA4F23C7 FPGAs that are constructed with 28nm process technology and availing maximum 250MHz clock speeds. Although the 35 gate path we picked is one of the long paths in ISCAS'85 benchmark, the resolution of 4ns ($1/250MHz$) is too high to precisely measure the delay of that path, which has a delay of about 8ns. Although, we roughly measure the delay of the path, it would be impossible to measure the extra delay caused by a small Trojan shown in (1). In order to be able to measure both path's and the Trojan's delay, we serially connect the multiple copies of the same path and then measure the delay of these serially connected paths, lastly divide the total delay to the number of copies to get the delay of the single path.

4. Effect of the Variations on Delay

The delay variations caused by manufacturing process are separated into two: the variations inside the same chip, i.e. intra-die variations and the variations among different chips of the same circuit, i.e. inter-die variations.

4.1. Intra Die Variations

In order to measure the intra-die delay variation component, we first put the same path on 8 different locations in the same FPGA chip as shown in Fig. 3. Each color corresponds to a different path. Each path actually consists of 50 copies of the same path connected in serial so that we can almost precisely measure their delays with 250MHz clock. The resultant path delays are shown in Table 1. The first row is the total measured delay for 50 serially connected path copies and the second row is the actual delay of the single path. This table reveals that the delay variation due to the intra-die component has $3\sigma/\mu$ ratio of 2.96%.

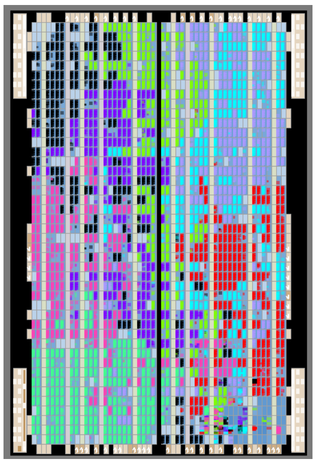

Fig. 3. Layout of 8 paths in different locations.

Table 1. Path Delays (in ns) in 8 different locations.

ID	L1	L2	L3	L4	L5	L6	L7	L8
50P	428	436	436	440	432	436	428	436
1P	8.56	8.72	8.72	8.8	8.64	8.72	8.56	8.72

[1]For AND, NAND gates 1; for OR, NOR gates 0.

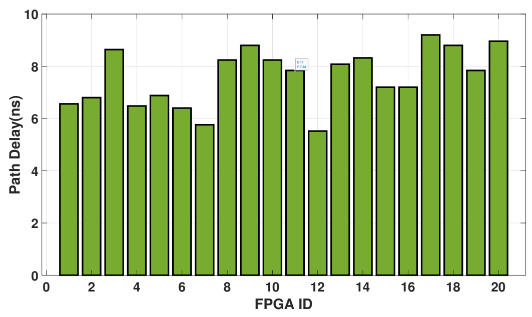

Fig. 4. Path Delays measured on 20 FPGAs.

4.2. Inter Die Variations

In order to measure the inter die variation component, the same path is uploaded to different FPGA chips and the delays are measured. We use 20 chips and the results are shown in Figure 4 as a bar plot. The path delays are in between 5ns and 10ns. The standard deviation (σ) and mean (μ) of this distribution result in a $3\sigma/\mu$ ratio of 43.5%, which is too far ahead of intra-die variations. One reason for that dramatic difference is that these FPGAs do not have to be from the same wafer or the same lot.

As the increase in the delay of the circuit can be either from the Trojan or the variation, such delay variations open up a room for hardware Trojans to hide inside.

5. Effect of the Hardware Trojan on Delay

The effect of the variation on actual 28nm FPGA chips is shown in the previous section. In this section, the effect of the inserted hardware Trojan is studied. For that purpose, we insert an XOR gate as the payload of the Trojan [7, 12] shown in Fig. 1. Of course, we insert the Trojan into each copy of the path under test for fair results.

Fig. 5 shows us 10 Trojan-free path delays and 10 Trojan-inserted path delays each corresponding to a different FPGA. This figure shows that a Trojan-inserted path can even have a much smaller delay than a Trojan-free path. For instance, FPGA-3 has a Trojan-free path delay of about 9ns whereas FPGA-12 has a Trojan-inserted path delay even smaller than 6ns. This shows that variations can perfectly hide the effect of the Trojan and that the detection looking solely at path delay is impossible as argued and supported with Spice simulation results in [10].

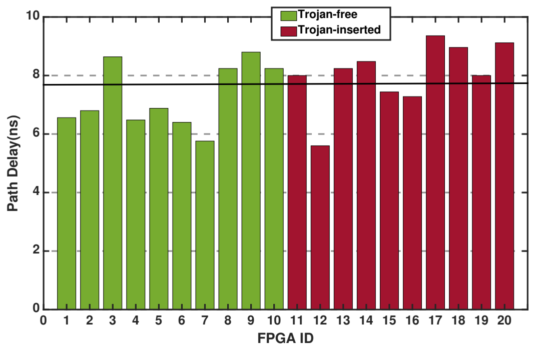

Fig. 5. Path delay measurements with HT added to half of the FPGAs.

Moreover, if we want to discriminate Trojan-inserted ones from the Trojan-free ones, even a carefully picked threshold as shown with a solid black line in Fig. 5 results in 3 false negatives and 4 false positives among 20 chips. False positives are labeled as Trojan-inserted although there is no Trojan and false negatives are the ones that are labeled as Trojan-free although they have Trojan. However, the tester is unable to pick such a threshold and even may assume all chips are Trojan-free as their delays are intermixed into each other and the effect of Trojan is totally hidden under the process variations. In order to handle that problem the effect of the variations must be suppressed, which is explained next.

Please note that throughout the paper the Trojan-free chips are shown by green bars and Trojan-inserted chips are shown by red bars.

6. Ring Oscillator to Suppress Variations

A ring oscillator (Fig. 6) is composed of odd number of serially connected inverters and used to determine the delay by solely looking at the frequency of its output, as its output continuously triggers with a period two times the delay of the ring oscillator (RO).

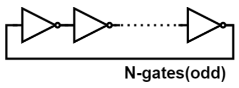

Fig. 6. A generic ring oscillator circuit.

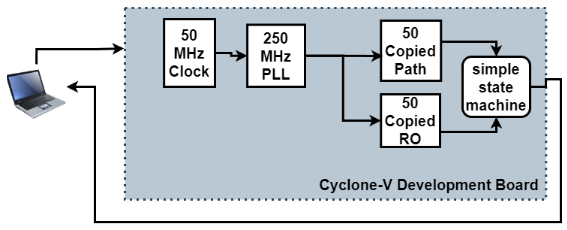

Fig. 7. Overview of the experimental setup.

We use ring oscillators with 35 inverters to give an idea about the delay of a specific FPGA chip. In other words, it is utilized to represent how the chip, that it is located in, is affected from the variations. Therefore, we put an additional ring oscillator to each of the 20 FPGA chips and then measure their delays. The experimental setup is shown in Fig. 7 and the FPGA layout after occupying the circuit is shown in Fig. 8.

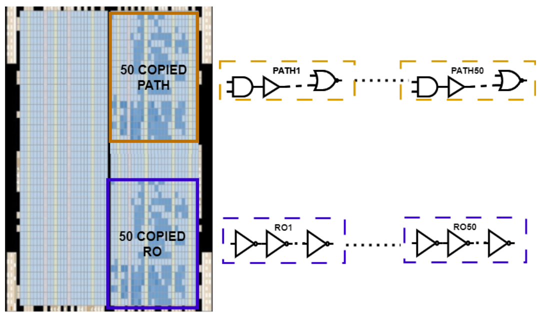

Fig. 8. FPGA Layout of the 50 copied path & ring oscillator.

As the ring oscillator is affected similarly from the inter die variations, a division of path delay to the ring oscillator delay for each chip can yield a variation cancellation. Therefore, for each chip we divide delay of the path under test to the delay of the ring oscillator. The resultant 20 delay ratios are shown in Fig. 9.

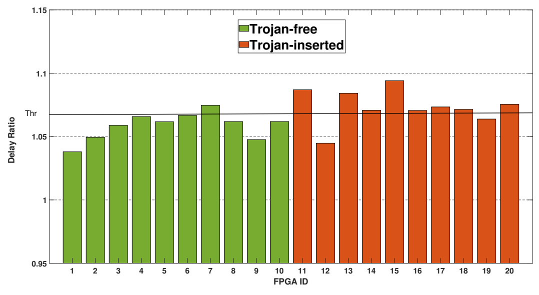

Fig. 9. Path delay ratios with HT added to half of the FPGAs.

Actually, the resultant delay ratios corresponding to the Trojan-inserted chips tend to be larger than the Trojan-free chips. Yet, it is still very difficult to distinguish the chips. For instance, if we put an almost perfect threshold shown as a black solid line in Fig. 9, then we still have 1 false positive and 2 false negatives among 20 chips, which is somewhat better than Fig. 5 results, where path delay is used alone.

We repeat the same experiment, but this time instead of the last 10 FPGAs, we insert the Trojan to the first 5 FPGAs. The resultant ratio graph is shown in Fig. 10. Again even a very well picked threshold shown as a solid black line results in 6 false positives.

Moreover, the tester in the actual case can not see the colors of the bars. Only the 20 delay ratio computations can be collected. Therefore, the threshold value to be used to distinguish Trojan-inserted chips from Trojan-free ones is unknown.

One way could be to test whether the distribution of the delay ratio measurements is bimodal. Because if the effect of the Trojan is high enough, the Trojan-free and Trojan-inserted chips would result in delay ratio distributions clustered in two distinct centers. This means there is an inserted Trojan in some of the chips. However, Hartigan's bimodality test [13] does not return a bimodal distribution meaning that it cannot understand that among the chips some of them are Trojan-inserted.

7. Size of the Trojan

The size of the Trojan and amount of variations are the main causes for that result. Because, the Trojan size with respect to the path size in these experiments is about 2.7%, which is smaller than the intra die variations. Using a larger Trojan payload than Fig. 1 can yield much better detection scenarios and bimodality results at the end of Hartigan's tests.

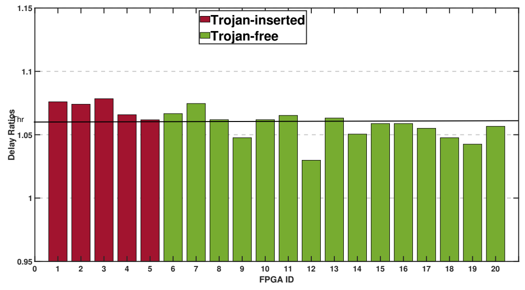

Fig. 10. Path delay ratios with HT added to quarter of FPGAs.

For instance, if we use two serial XORs instead of one the resultant delay ratios are as shown in Fig. 11. The result is getting better if the tester can pick the threshold shown in the figure. But still there are two false negatives. On the other hand when we increase the number of serial XORs up to three, the resultant delay ratios can finally provide a perfect separation.

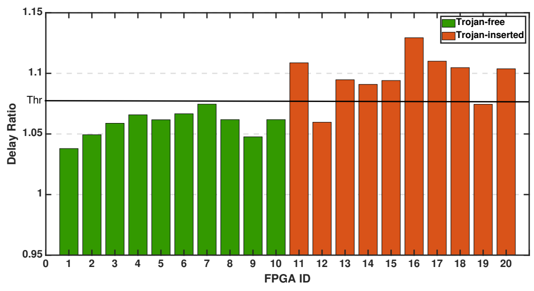

Fig. 11. Two serial XORs are used as Trojan

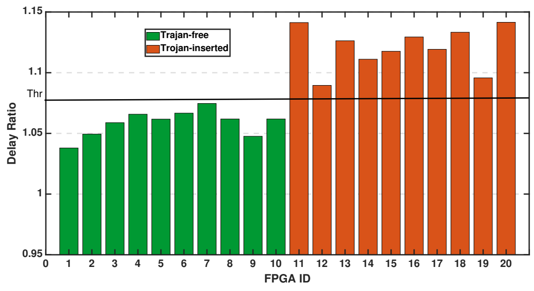

Fig. 12. Three serial XORs are used as Trojan

When we check the distributions of delay ratios coming from 20 chips, the resultant histograms for 1 XOR, 2 XOR, 3 XOR and 5 XOR Trojan cases are shown as HTx1, HTx2, HTx3 and HTx5 respectively in Fig. 13. It can be deduced from the figure that the bimodality of the resultant delay ratio distribution is increasing with the increasing HT size, which is also verified by Hartigan results. Because, the `p values` given by the Hartigan's test are shown in Table 2, where `p values` smaller than 0.1 shows the bimodality of the distribution [10]. As a result, for HTx5 case, the bimodality and therefore the Trojan is detected and thus, a threshold can easily be found as the middle of two distributions to distinguish Trojan-inserted circuits.

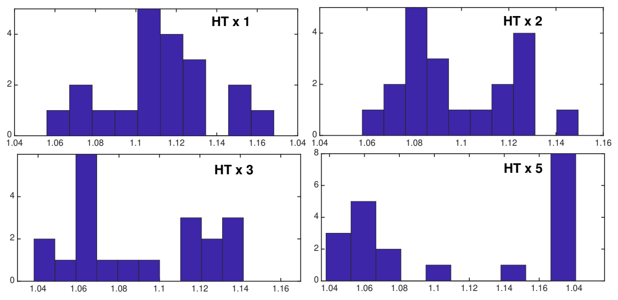

Fig. 13. Delay ratio histograms for different Trojan sizes

Another bimodality test in the literature is Warren Sarle's bimodality test [14], in which bimodality coefficients larger than 0.55 means a bimodal or multimodal distribution and lower than that value mean a unimodal distribution. When the distributions of Fig. 13 are given to Sarle's test, the results are in Table 2. Similar to the dip test, the results are getting much closer to a bimodal distribution while the Trojan size increases but only the HTx5 case is labeled as bimodal.

Table 2. Bimodality Test Results

Bimodality Test	**HTx1**	**HTx2**	**HTx3**	**HTx5**
Hartigan	0.61	0.35	0.21	0
Sarle	0.29	0.42	0.52	0.66

However, even a payload of a single XOR gate (HTx1), is enough to hide Trojan from logic tests and to change the functionality of the circuit. Therefore, a detection scheme must take this into account and find more sophisticated methods to get rid of the variations as in [10].

8. Conclusions

From the measurements on 20 FPGA chips, it can be concluded that firstly, the effect of delay variations is overwhelming, especially the inter die component and that the variations can easily hide the effect of hardware Trojans with payloads as small as one XOR gate.

Even if we insert an additional ring oscillator to the circuit in order to capture the variations, we still can not detect the Trojans as the resultant delay ratios pass from Hartigan's test as a unimodal distribution, which means that the resultant delays seem to be clustered in one center and which means that inserted Trojans are not detected although half of the chips have inserted Trojans. The detection scheme is getting better with the increasing Trojan size. However, the adversary does not need large payloads for hiding and constituting the Trojan. This shows the necessity of sophisticated delay based detection methods to suppress the variations and reveal the effect of hardware Trojan in the circuit.

9. References

[1] M. M. Tehranipoor and F. Koushanfar, "A survey of hardware trojan taxonomy and detection," *IEEE design & test of computers*, vol. 27, no. 1, pp. 10–25, 2010.

[2] S. Bhunia and M. M. Tehranipoor, "The Hardware Trojan War," *Cham,, Switzerland: Springer*, 2018.

[3] A. P. Fournaris, L. Pyrgas, and P. Kitsos, "An efficient multiparameter approach for fpga hardware trojan detection," *Microprocessors and Microsystems*, vol. 71, p. 102863, 2019.

[4] F. N. Esirci and A. A. Bayrakci, "Hardware trojan detection based on correlated path delays in defiance of variations with spatial correlations," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*. IEEE, 2017, pp. 163–168.

[5] J. Li and J. Lach, "At-speed delay characterization for IC authentication and Trojan horse detection," in *IEEE International Workshop on Hardware-Oriented Security and Trust*. IEEE, 2008, pp. 8–14.

[6] N. Yoshimizu, "Hardware Trojan detection by symmetry breaking in path delays," in *IEEE International Symposium on Hardware-Oriented Security and Trust*. IEEE, 2014, pp. 107–111.

[7] X. Cui, E. Koopahi, K. Wu, and R. Karri, "Hardware Trojan detection using the order of path delay," *ACM Journal on Emerging Technologies in Computing Systems*, vol. 14, no. 3, pp. 1–23, 2018.

[8] A. Vakil, A. Mirzaeian, H. Homayoun, N. Karimi, and A. Sasan, "AVATAR: NN-Assisted Variation Aware Timing Analysis and Reporting for Hardware Trojan Detection," *IEEE Access*, vol. 9, pp. 92 881–92 900, 2021.

[9] H. E. Taheri and M. Mirhassani, "A Pre-Activation, Golden IC Free, Hardware Trojan Detection Approach," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 30, no. 3, pp. 315 – 324, 2022.

[10] F. N. Esirci and A. A. Bayrakci, "Delay based hardware trojan detection exploiting spatial correlations to suppress variations," *Integration*, vol. 91, pp. 107–118, 2023.

[11] F. Brglez and H. Fujiwara, "A neutral netlist of 10 combinational benchmark circuits and a target translator into fortran," in *IEEE Int'l Symp. on Circuits and Systems (ISCAS)*, 1985, pp. 659–662.

[12] M. T. Hassan Salmani, "AES-T2300/2400 from Trust-Hub." 2013.

[13] J. A. Hartigan and P. M. Hartigan, "The dip test of unimodality," *The annals of Statistics*, vol. 13, no. 1, pp. 70–84, 1985.

[14] R. Pfister, K. Schwarz, M. Janczyk, R. Dale, and J. Freeman, "Good things peak in pairs: a note on the bimodality coefficient," *Frontiers in Psychology*, vol. 4, 2013.