Adversarial Hardware With Functional and Topological Camouflage

He Li, *Member, IEEE*, Ameer Abdelhadi, *Member, IEEE*, Runbin Shi, Jiliang Zhang, *Senior Member, IEEE*, and Qiang Liu, *Member, IEEE*

***Abstract*—The lately attack on the Qualcomm's Snapdragon chip reminds us security risks related to digital arithmetic circuits. Most system designers neglect the employed computer arithmetic algorithms or their implementation details. Arithmetic circuits are therefore usually used as "black box" units that are instantiated by third-party intellectual-property core or electronic design-automation tool vendors. In this brief, we propose the first most-significant digit-first arithmetic-based hardware Trojan attack by addressing the questions of *where* to insert and with *what*. First, we demonstrate how *functional* camouflage is achieved. Certain arithmetic modules can be quietly replaced with functionally equivalent ones, which we term functionally camouflaged Trojans. Next, we introduce a *topologically* camouflaged Trojan by employing graph-centrality analysis on rare behaviours in the circuit. The proposed approach is applicable to any digital computing scenarios. Experimental results on a financial computing system demonstrate that completely inaccurate numeric results are yielded, along with an up-to 91.6% numerical error tested.**

***Index Terms*—Hardware security, MSDF arithmetic, graph analysis.**

I. Introduction

GROWING interest in the hardware acceleration of applications including finance [1], machine learning [2] and security [3] is causing frequent reconsideration of the implementation of basic arithmetic operators. While some operations, notably division, produce their output digits in order of decreasing significance, this is not true of the operations most often found in such applications: addition and multiplication. The unified most-significant digi-first (MSDF) computation promises deep pipeline parallelism, graceful degradation, and trivial support for customized variable-precision operation. Therefore, MSDF or "left-to-right," arithmetic has recently returned to prominence due to its attractive properties for field-programmable gate array (FPGA) and application-specific integrated circuit (ASIC) implementation [4].

With the scaling of digital computing systems, the growing demand of arithmetic circuits relies on third parties to provide the arithmetic modules, thereby resulting in potential security threats [5]. In August 2020, a research team named "Achilles" reported over 400 vulnerabilities within the digital signal processor (DSP) in Qualcomm's Snapdragon chip, affecting over 40% of the global mobile phone market [6]. Due to the "black box" nature of the outsourced DSP chips, it is challenging for the mobile vendors to fix these issues [6]. This attack alerts security threats arising from external intellectual property (IP) cores or related electronic design automation (EDA) tools, as they are completely beyond the control of system designers [7], [8]. The security vulnerabilities in Qualcomm's Snapdragon DSPs indicate that system designers usually have no knowledge about the design details of arithmetic circuits. Use of the inherent logic of different computer arithmetic algorithms can allow us to design a *functionally* camouflaged adversarial hardware Trojan (HT). Herein, distinguished MSDF arithmetic and conventional least-significant digit-first (LSDF) arithmetic are employed. With the same functionality, MSDF arithmetic operation provides a natural camouflage against its LSDF equivalence for attackers, and vice versa.

As a widely discussed hardware attack scheme, HTs are usually designed at *rare* conditions or signals in a circuit [9]. We first investigate how to model the rare behaviour within a circuit, which is the key to HT insertion. Rare circuit signals have low 0-1 transition activities, therefore, they have negligible effect on the rest of the circuit [7]. In graph theory, *betweenness centrality* has been used to measure the influence of a signal in digital circuits [10]. The lower the betweenness centrality is, the less influential the signal will be [10]. We utilized this analysis to predict rare signals in the circuits, thereby producing a *topological*-camouflage HT insertion. Combining functional with topological camouflage, we propose a novel attack method to insert arithmetic-based HTs at low-betweenness locations. The contributions of this brief are listed as follows:

Manuscript received February 5, 2021; accepted March 8, 2021. Date of publication March 10, 2021; date of current version April 30, 2021. This work was supported in part by the National Natural Science Foundation of China under Grant U20A20202, Grant 61974102, and Grant 61874042; in part by the Tianjin Municipal Transportation Science and Technology Development Plan Project under Grant 2017B-40; in part by the Hunan Natural Science Foundation for Distinguished Young Scholars under Grant 2020JJ2010; and in part by the Hu-Xiang Youth Talent Program under Grant 2018RS3041. This brief was recommended by Associate Editor F. Pareschi. *(Corresponding authors: Jiliang Zhang; Qiang Liu.)*

He Li is with the Department of Engineering, University of Cambridge, Cambridge CB2 1PZ, U.K. (e-mail: hl556@cam.ac.uk).

Ameer Abdelhadi is with the Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON M5S 3G4, Canada.

Runbin Shi is with the Department of Electrical and Electronic Engineering, University of Hong Kong, Hong Kong.

Jiliang Zhang is with the College of Computer Science and Electronic Engineering, Hunan University, Changsha 410082, China (e-mail: zhangjiliang@hnu.edu.cn).

Qiang Liu is with the School of Microelectronics, Tianjin University, Tianjin 300072, China (e-mail: qiangliu@tju.edu.cn).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCSII.2021.3065292.

Digital Object Identifier 10.1109/TCSII.2021.3065292

- *Functional* camouflage for HT insertion is realised by exploiting theoretical differences between MSDF and LSDF arithmetic algorithms.
- *Topological* camouflage for HT insertion is achieved by employing graph-centrality analysis, where HTs can be inserted at locations with low-centrality values.
- Attack evaluation on a security-critical financial computing for cash-flow analysis is presented, where datapaths built upon our principle pruduce completely inaccurate results. Detailed robustness analyses against state-of-the-art HT detection methods are also discussed.

The implementations evaluated in this brief target security-critical applications. Our principles are, however, generic, and can be employed to any digital computing systems.

II. Background

A. Hardware Trojan Design

HTs are modifications to the original circuit by adversaries to gain access to confidential information or to destroy its functionality. A typical model of an HT contains a trigger and a payload [7]. The trigger is usually associated with rare signals or conditions. When an HT is triggered, the payload circuit results in malicious functions. By careful design, the rare signals or conditions are unlikely to arise during simulation or testing but can occur over long periods of field operation [9]. Generally, HTs can be categorised into two types: combinational Trojan and sequential Trojan [11]. A combinational Trojan is triggered by the simultaneous occurrence of a set of rare signals. A sequential Trojan undergoes a sequence of rare events, each triggered by a different set of rare signals, before activating the payload. Considering a signal that attackers expect to modify its value with an HT, at least one dedicated trigger input must be employed to activate this HT circuit. Therefore, the key to insert HTs is to identify the rare signals or rare events in circuits.

After reviewing recent HT designs and implementations [7], [9], and the widely used Trust_hub with 93 HT-inserted benchmarks [12], arithmetic-based HTs have not been well investigated. As a lessen learnt from Qualcomm's Snapdragon chip [6], it is important to figure out how arithmetic algorithms can be used for hardware attack in practical scenarios. Even though few HT modifications to basic arithmetic elements have been studied [13], [14], we are the first employing the algorithm natures of MSDF and LSDF arithmetic for HT designs.

B. Verification-Based HT Detection and Limitations

Since HTs designed by adversaries are resistant to traditional functional verification approaches, in recent years, several IP trust verification techniques have been proposed to prevent hardware designs from HT attacks. FANCI has been proposed to discover HTs in hardware designs using functional Boolean analysis [15]. The approach identifies nearly-unused logic using a metric that measures the control ability of each input in a digital circuit. Similar to FANCI, Hicks *et al.* formulated the HT detection problem as an unused circuit identification (UCI) problem [16]. A piece of circuit is considered suspicious if it does not affect any primary outputs during testing. However, the approach could only cover a small set of HTs due to the relatively simple definition of unused circuit [17]. An optimized approach, VeriTrust, was presented to identify potential trigger inputs of parasite-based HTs [17]. Zhang and Tehranipoor defined all functions and corresponding assertions in the specification. Once these are completed, coverage metrics (code coverage and functional coverage) are used in authentication to help identify suspicious parts in third-party IP cores [18]. Unfortunately, the aforementioned methods are time-consuming, usually taking several hours or even days, thereby only practical for small-scale circuits and systems. The aggregation of component verification does not equate formal verification of the entire integrated system [19]. Zou *et al.* demonstrated that verification-based methods have a time complexity of $\mathcal{O}(N_{\text{en}}N_{\text{net}}2^{N_{\text{rec}}})$, where N_{en} is the number of entries of a state machine, N_{net} is the number of nets, and N_{rec} denotes the number of reconvergent inputs of a circuit under test [20]. With an increasing size of modern designs, the difficulty to identify HTs increases exponentially.

Interests in recent HT detection based on feature analysis [21], and those employing machine learning algorithms [22], have arisen recently. However, these methods rely heavily on the available dataset of HTs, such as Trust_hub [12]. Therefore, these methods are not well-equipped to detect the proposed functionally camouflaged HTs, considering that defenders have no access to our MSDF arithmetic operators.

C. Most-Significant Digit-First Arithmetic

Arithmetic algorithms have two typical operation modes, i.e., LSDF and MSDF [23]. In LSDF computing, operand and result digits are applied from the least-significant end, such as traditional addition and multiplication. MSDF computing instead applies operand and result digits from the most-significant end. A *de facto* standard for MSDF arithmetic is *online arithmetic* [23]. By prioritizing the production of more important data, applications can benefit from throughput, latency, and energy efficiency improvements. Online operators are classically serial, however efficient digit-parallel (unrolled) implementations targeting FPGAs have been developed [23]. Users can choose both digit-serial and -parallel online operators in their design. For example, digit-serial online adder is presented in Figure 1 (left), while duplication of the serial adder M times and the removal of its registers lead to the creation of a M-digit parallel online adder, as shown in Figure 1 [23]. Even though carrys are delivered at the LSD end and generated at the MSD end in online adders, there is no carry chain; the critical path lies across two full adders (FAs) [23]. This implies online adder's suitability for the construction of more complex online operators, such as multiplication and division.

Of particular significance to the proposed functional camouflage is the concept of *online delay*. When performing an MSD-first operation, the digits of its result are generated at the same rate as its input digits are consumed, but the result

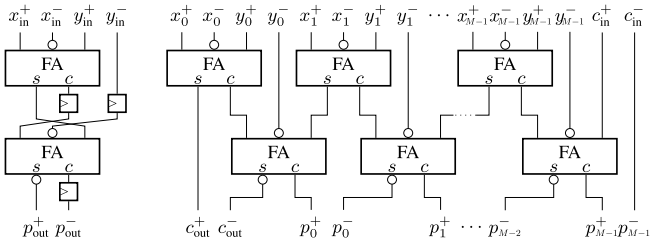

Fig. 1. Radix-2 online adders. Left: serial. Right: parallel [4].

is delayed by a fixed number of digits, denoted δ. That is, the first most-significant m digits of an operator's result are wholly determined by the first $m+\delta$ digits within each of its operands. The value of δ is operation-specific, which is a small integer (1 to 4) determined by the redundancy factor and the radix [23].

III. The Proposed Camouflaged Hardware Trojan

A. Threat Model

Our threat model is based on the model discussed in [5]. We assume three parties are involved: (i) a *client* that wants a *designer* to build digital computing systems in security-critical domains, without revealing the confidential data; (ii) a *benign designer* that develops designs using third-party EDA tools or IP cores; (iii) an *untrustworthy vendor* that provides third-party EDA tools or IP cores to the *designer* [5]. We also assume attackers have knowledge of arithmetic algorithms to make functional-equivalent arithmetic cores harder to discriminate. The goal of attackers is to control exactly *where* and *what* to insert, which have minimal impact on overall overhead. Consider a scenario that a simple arithmetic operation is deployed in very-large scale systems. Functional camouflage can be achieved by substituting MSDF and LSDF arithmetic IP cores, resulting in completely inaccurate computed results and system malfunction. Given a high-level functional description from clients, topological camouflage can be realised via graph centrality with a shortlist of rare signals in a circuit. Therefore, the benign designer is difficult to prevent the functional camouflaged replacement, even though hardware implementations of MSDF and LSDF arithmetic operations are different.

B. Functional Camouflage

Since third-party IP cores and EDA tools are widely used in the standard flow of digital circuit design, it is possible for attackers to deliberately replace a conventional LSDF arithmetic core in the datapath with an MSDF one, and vice versa. We employ the inherent logic between functionally-equivalent arithmetic cores to design a functionally camouflaged HT, which can calculate completely wrong results as its payload.

As a standard practice in digital system design nowadays, various arithmetic IPs, either LSDF or MSDF, are usually used from third-party vendors. For LSDF arithmetic, fixed-point or floating-point, one's or two's complement representations are popular number systems implemented. When using such number systems, overflow detection plays a significant role in each arithmetic circuit [23]. To eliminate overflow, traditional two's-complement LSDF fixed-point arithmetic restricts its p-bit outputs within $[-2^{p-1}, 2^{p-1}-1]$. Instead, MSDF arithmetic employs a redundant number system and its digit-computation dependency (i.e., overflow detection logic) is completely different from LSDF equivalents. Use of the same radix, a p-digit MSDF operation allows the results to be represented in the interval of $[-2^{-p}, 2^{-p}]$. Therefore, an MSDF arithmetic operator performs the same functionality of its LSDF equivalence, but can calculate several additional values. Attackers can employ this fundamental difference in arithmetic algorithms to launch an functional-camouflaged HT attack. Since this attack is introduced within the operator itself, therefore it is more camouflaged than a classic hardware Trojan [13], [14].

C. Topological Camouflage

We now investigate the topological camouflage by using betweenness centrality which can measure the relative importance of a node/edge in a graph [24]. Since HT triggers are related to rare signals, such signals should be less important and even ignorable in a circuit. In this brief, we employed connections between the rareness of HT triggers and low betweenness-centrality signals to explore possible locations for HT insertion.

1) Betweenness Centrality: Let a graph $G(V, E)$ contain a set of vertices V and a set of edges E. Edge $e_{ij} \in E$ is directed from vertex $v_i \in V$ to vertex $v_j \in V$. The betweenness centrality $C_B(i, j)$ of an edge $e_{ij} \in E$ is defined as

$$C_B(i,j) = \sum_{k,h} \frac{|e_{k,h,(i,j)}|}{|e_{kh}|}, \quad k \neq h, \quad k, h \in V. \tag{1}$$

$|e_{k,h,(i,j)}|$ is the number of shortest paths from node k to node h going through edge e_{ij}, and $|e_{kh}|$ is the number of shortest paths between k and h. Given the path length and the shortest paths count, $\frac{|e_{k,h,(i,j)}|}{|e_{kh}|}$ is the pair-dependency of a pair k, $h \in V$ on an intermediary edge $e_{ij} \in E$. The time complexity of betweenness centrality is $\mathcal{O}(n_{\mathrm{v}} n_{\mathrm{e}})$, where n_{v} and n_{e} are the number of vertices and edges in a graph [24].

Our calculation is based on an adjacency matrix that represents a directed graph extracted from a Verilog or netlist file. Vertices are used to represent primitives of a hardware design and edges indicate a connection between vertex pairs. In the following, we determine the betweenness centrality for each node (edge) in two steps: 1) calculating the path length and the number of shortest paths between all pairs of vertices; 2) accumulating all pair-dependencies, as will be demonstrated in Section IV.

2) Rare Signal Identification: After the computation of betweenness centrality, signals associated with the nodes/edges that have the smallest betweenness centrality values are shortlisted. Algorithm 1 describes how rare signals are flagged within an untrusted design in our approach. Use of the generated connectivity graph enables to compute and rank the betweenness-centrality values of each node and each edge. If a centrality value is smaller than the threshold ε, the related part in the design is flagged as a rare node. Herein, the threshold value can be chosen either a priori or after looking at the distribution of computed betweenness centrality values. We

Algorithm 1 Flag Rare Signals in a Design

Input: Connectivity graph $G(V, E)$ of a circuit and a threshold ε.
Output: A predicted rare-signal set S.

```
S ⇐ An empty set of rare signals
for all edges e(i, j) ∈ E do
    Compute betweenness centrality C_B(i, j)
    if C_B(i, j) ≤ ε then
        S ⇐ e_ij
    end if
end for
```

remark that the latter option is preferable if attackers are able to know such distribution. In any case, the chosen threshold helps recommend rare signals for HT insertion to gain topological camouflage.

IV. Experimental Results

A. Attack on Financial Computing

In order to evaluate the proposed camouflaged attack, we implemented a widely used financial computing algorithm—cash-flow analysis for the calculation of interest rate—following the aforementioned principles. We chose cash-flow analysis to exemplify a large class of security-critical digital computing systems with the use of arithmetic operations. In reality, cash-flow analysis usually contains confidential data, such as history payments of n successive periods, defined as $c_0, c_1, \ldots, c_{n-1}$. The interest rate x can be obtained by finding the root of a polynomial characterized as

$$f(x) = c_0 + c_1 x + c_2 x^2 + \cdots + c_{n-2} x^{n-2} + c_{n-1} x^{n-1}. \quad (2)$$

As widely discussed in [4], Newton's method is used to solve the polynomial $f(x)$ with coefficients $c_i, i \in [1, n-1]$ and initial investment c_0.

According to the threat model in Section III-A, a bank client, a designer and an attacker are involved in this financial case. To implement this cash-flow analysis characterized in (2), a designer will develop a datapath such as that shown in Fig. 2. Since we assume the designer replies on untrustworthy tools and IP cores provided by the attacker, Fig. 2 can be converted to a graph and betweenness-centrality values for each node are calculated. Thereafter, LSDF arithmetic circuits with low betweenness centrality are replaced with their MSDF equivalents. For example, four rare arithmetic cores are highlighted in Fig. 2, and one with $C_B = 0.0189$ is used for functional camouflage attack. In real-life scenarios, the bank client has to protect these confidential payment coefficients [25], therefore, a *warning* function must be requested in the cash-flow analysis design in order to alert an extreme payment event. Unfortunately, with our camouflaged HT (i.e., an MSDF adder in Fig. 2), the hacked design can bypass the warning and send a wrong computed results directly.

B. Attack Evaluation

To demonstrate the effect of the proposed attack scheme, we recall the exemplary sets of real-life payment coefficients [25],

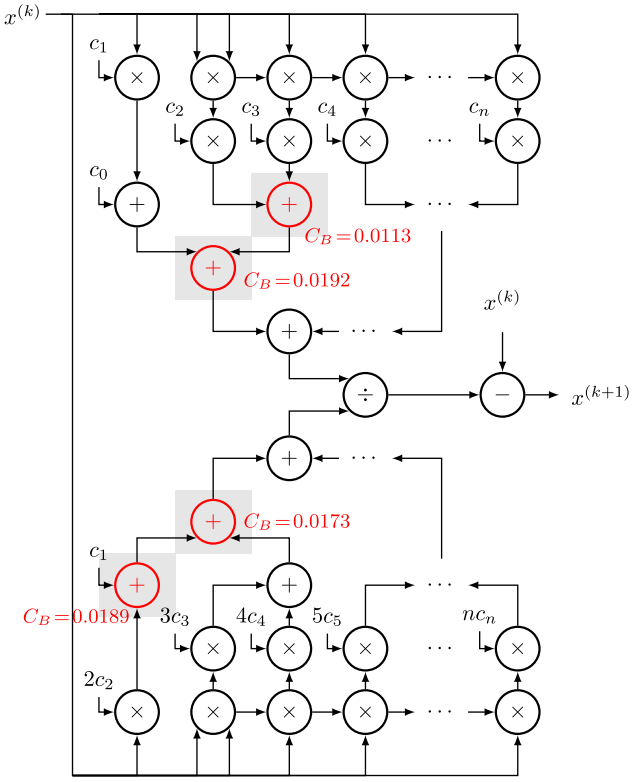

Fig. 2. A datapath of the cash-flow analysis. Red LSDF adders are selected by our graph-centrality analysis with $\varepsilon = 0.0200$, where one with $C_B = 0.0189$ are maliciously hacked as MSDF adder.

TABLE I
Exemplary Sets of Cash-Flow Coefficients

Payments	c_0	c_1	c_2	c_3	c_4	c_5
Case 1	-2016.0	6921.9	3133.2	1008.5	51.1	132.6
Case 2	-17333	16567	8471	7432	6812	6571
Payments	c_6	c_7	c_8	c_9	c_{10}	c_{11}
Case 1	87.6	-34.9	-10.4	-2.4	-0.5	-0.1
Case 2	-6739	5538	4871	-4372	-3970	-3658

TABLE II
Interest Rate of Cash Flow Analysis Using Case 2 in Table I

# Iterations	Displayed results	Requested results	True results
1	1.046	Warning (overflow)	0.4048
2	1.769	Warning (overflow)	0.6930
3	1.612	Warning (overflow)	0.6367
4	1.476	Warning (overflow)	0.6335
5	1.365	Warning (overflow)	0.6335
6	1.283	Warning (overflow)	0.6335
7	1.235	Warning (overflow)	0.6335
8	1.214	Warning (overflow)	0.6335
Error	✔ (91.6%)	✘	✘

$c_i, i \in [0, 11]$, as shown in Table I. The computation of interest rates starts with an initial guess $x^{(0)} = 0$. Table II shows the computed interest rate using our HT-inserted design in terms of *displayed*, *true*, and *requested* results. When c_is correspond to usual payment situations (e.g., Case 1), the hacked design functions well, showing the correct interest rate. When an extreme payment event (e.g., Case 2) occurs, c_1, c_2 and c_3 with consecutively large values should result in an overflow warning. However, our camouflaged attack has quietly

replaced the LSDF adder with a functional-equivalent MSDF adder. Given that an MSDF adder has a larger overflow tolerance, the hacked design has to continue calculating wrong interest rates as 0.6335, while the true result is 1.214. Since the bank client usually has no knowledge of the implementation details, this wrong result is likely to be used in trading, leading to potential economic losses. MSDF adder requires a small amount of extra area, but as was elaborated in Section II-C, this overhead is small and amortised out the more operators are instantiated for larger-scale systems. For hardware acceleration of applications, e.g., MSDF signal processing, or machine learning (ML) inferences [4], we can thus replace MSDF with LSDF arithmetic cores, leading to no area overhead.

C. Robustness Analysis

The most relevant HT detection method is Li *et al.*'s proposal for activating HTs in DSP circuits using signal statistical properties [11]. Since MSD- and LSD-first arithmetic operations are functionally-equivalent, they have the same statistical properties. Li *et al.*'s HT activation method is unfortunately unable to detect our attacks. Verification-based HT detection suffers from long testing time with a time complexity of $\mathcal{O}(N_{\text{en}}N_{\text{net}}2^{N_{\text{rec}}})$ [20], as was mentioned in Section II-B, therefore the detection time scales exponentially with the design complexity. Some other HT detection methods exploited HT structure features [21] and ML algorithms [22]. With HT samples such as ones in Trust_hub benchmarks, these methods showed satisfactory detection capability. However, defenders usually have no HT details or the golden standard reference, in particular of the MSDF arithmetic circuits, therefore defending against the proposed attack is non-trivial.

V. Conclusion

An adversarial HT design is proposed with functional and topological camouflages, by employing theoretical differences between arithmetic paradigms and graph-centrality analysis. Given the principle produced herein, our proposal is generic for any digital computing applications, such as machine learning and digital signal processing. Experimental results on cash-flow analysis in the security-critical financial computing domain demonstrate the effectiveness of our attack. In the future, we are interested to investigate functional camouflaged HT attack in designs using high-level synthesis (HLS) and approximate computing techniques. We will also investigate other centrality schemes in graph theory, since the more topological camouflage we provide, the more difficult an HT can be detected.

References

[1] K. Glau, D. Kressner, and F. Statti, "Low-rank tensor approximation for Chebyshev interpolation in parametric option pricing," *SIAM J. Financ. Math.*, vol. 11, no. 3, pp. 897–927, 2020.

[2] J. Zhang and G. Qu, "Physical unclonable function-based key sharing via machine learning for IoT security," *IEEE Trans. Ind. Electron.*, vol. 67, no. 8, pp. 7025–7033, Aug. 2020.

[3] J. W. Bos and S. J. Friedberger, "Arithmetic considerations for isogeny-based cryptography," *IEEE Trans. Comput.*, vol. 68, no. 7, pp. 979–990, Jul. 2019.

[4] H. Li, J. J. Davis, J. Wickerson, and G. A. Constantinides, "ARCHITECT: Arbitrary-precision hardware with digit elision for efficient iterative compute," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 28, no. 2, pp. 516–529, Feb. 2020.

[5] J. Zhang and G. Qu, "Recent attacks and defenses on FPGA-based systems," *ACM Trans. Reconfig. Technol.*, vol. 12, no. 3, pp. 1–24, 2019.

[6] *Achilles*. Accessed: Aug. 6, 2020. [Online]. Available: https://blog.checkpoint.com/2020/08/06/achilles-small-chip-big-peril/

[7] K. Xiao, D. Forte, Y. Jin, R. Karri, S. Bhunia, and M. Tehranipoor, "Hardware Trojans: Lessons learned after one decade of research," *ACM Trans. Design Autom. Electron. Syst.*, vol. 22, no. 1, pp. 1–23, 2016.

[8] A. M. S. Abdelhadi and M. R. Greenstreet, "Interleaved architectures for high-throughput synthesizable synchronization FIFOs," in *Proc. IEEE Int. Symp. Asynchronous Circuits Syst.*, 2017, pp. 41–48.

[9] M. Xue, C. Gu, W. Liu, S. Yu, and M. O'Neill, "Ten years of hardware Trojans: A survey from the attacker's perspective," *IET Comput. Digit. Techn.*, vol. 14, no. 6, pp. 231–246, 2020.

[10] E. Hung and S. J. E. Wilton, "Scalable signal selection for post-silicon debug," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 21, no. 6, pp. 1103–1115, Jun. 2013.

[11] H. Li, Q. Liu, and F. Chen, "Signal word-level statistical properties-based activation approach for hardware Trojan detection in DSP circuits," *IET Comput. Digit. Techn.*, vol. 12, no. 6, pp. 258–267, 2018.

[12] H. Salmani, M. Tehranipoor, and R. Karri, "On design vulnerability analysis and trust benchmarks development," in *Proc. IEEE Int. Conf. Comput. Design*, 2013, pp. 471–474.

[13] S. Ghandali, G. T. Becker, D. Holcomb, and C. Paar, "A design methodology for stealthy parametric Trojans and its application to bug attacks," in *Proc. Int. Conf. Cryptogr. Hardw. Embedded Syst.*, 2016, pp. 625–647.

[14] J. Clements and Y. Lao, "Hardware Trojan design on neural networks," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2019, pp. 1–5.

[15] A. Waksman, M. Suozzo, and S. Sethumadhavan, "FANCI: Identification of stealthy malicious logic using boolean functional analysis," in *Proc. ACM Conf. Comput. Commun. Security*, 2013, pp. 697–708.

[16] M. Hicks, M. Finnicum, S. T. King, M. M. K. Martin, and J. M. Smith, "Overcoming an untrusted computing base: Detecting and removing malicious hardware automatically," in *Proc. IEEE Symp. Security Privacy*, 2010, pp. 159–172.

[17] J. Zhang, F. Yuan, L. Wei, Y. Liu, and Q. Xu, "VeriTrust: Verification for hardware trust," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 34, no. 7, pp. 1148–1161, Jul. 2015.

[18] X. Zhang and M. Tehranipoor, "Case study: Detecting hardware Trojans in third-party digital IP cores," in *Proc. IEEE Int. Symp. Hardw.-Oriented Security Trust*, 2011, pp. 67–70.

[19] J. Zhang, F. Yuan, and Q. Xu, "Detrust: Defeating hardware trust verification with stealthy implicitly-triggered hardware Trojans," in *Proc. ACM Conf. Comput. Commun. Security*, 2014, pp. 153–166.

[20] M. Zou, X. Cui, L. Shi, and K. Wu, "Potential trigger detection for hardware Trojans," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 7, pp. 1384–1395, Jul. 2018.

[21] A. Vijayan, M. B. Tahoori, and K. Chakrabarty, "Runtime identification of hardware Trojans by feature analysis on gate-level unstructured data and anomaly detection," *ACM Trans. Design Autom. Electron. Syst.*, vol. 25, no. 4, pp. 1–23, 2020.

[22] J. R. Hamlet, J. R. Mayo, and V. G. Kammler, "Targeted modification of hardware Trojans," *J. Hardw. Syst. Security*, vol. 3, no. 2, pp. 189–197, 2019.

[23] M. D. Ercegovac and T. Lang, *Digital Arithmetic*. Amsterdam, The Netherlands: Elsevier, 2004.

[24] U. Brandes, "A faster algorithm for betweenness centrality," *J. Math. Soc.*, vol. 25, no. 2, pp. 163–177, 2001.

[25] L. Thorlund-Petersen, "Global convergence of Newton's method on an interval," *Math. Methods Oper. Res.*, vol. 59, no. 1, pp. 91–110, 2004.