Hiding in Plain Sight: Reframing Hardware Trojan Benchmarking as a Hide&Seek Modification

Amin Sarihi, Ahmad Patooghy, *Senior Member, IEEE*, Peter Jamieson, and Abdel-Hameed A. Badawy, *Senior Member, IEEE*

***Abstract*—This letter focuses on advancing security research in the hardware design space by formally defining the realistic problem of hardware Trojan (HT) detection. The goal is to model HT detection more closely to the real world, i.e., describing the problem as "The Seeker's Dilemma" where a detecting agent is unaware of whether circuits are infected by HTs or not. Using this theoretical problem formulation, we create a benchmark that consists of a mixture of HT-free and HT-infected restructured circuits while preserving their original functionalities. The restructured circuits are randomly infected by HTs, causing a situation where the defender is uncertain if a circuit is infected or not. We believe that our innovative benchmark and methodology of creating benchmarks will help the community judge the detection quality of different methods by comparing their success rates in circuit classification. We use our developed benchmark to evaluate three state-of-the-art HT detection tools to show baseline results for this approach. We use principal component analysis to assess the strength of our benchmark, where we observe that some restructured HT-infected circuits are mapped closely to HT-free circuits, leading to significant label misclassification by detectors.**

***Index Terms*—Benchmark, hardware Trojan (HT), machine learning (ML), netlist.**

I. Introduction

HARDWARE Trojans (HTs) are a threat to digital electronics in general and the design and manufacturing of integrated circuits (ICs), in particular [1]. HTs are unwanted modifications in the design or manufacturing of an IC such that the chip's expected behavior is altered. Potential impacts of security breaches through HTs have pushed researchers to look into the methods and algorithms to detect HTs in ICs in the early stages of the design and manufacturing [2], [3], [4], [5], [6]. Despite the novelty, the field still needs a formal definition of the problem that mirrors the real-world problem of HT detection.

In this letter, our goal is to advance the state-of-the-art in HT detection by defining the problem of HT insertion/detection in digital ICs as a game between two players. Our problem statement is rooted in the *Hide&Seek* problem on a graph. We call this new formulation "The Seeker's Dilemma" as it more closely resembles the problem of HT detection from the perspective of real-world manufacturers and distributors of ICs. We take The Seeker's Dilemma approach in creating realistic HT benchmarks to address the shortcomings of existing HT benchmarks. In current benchmarks, the HT location and size are known a priori to researchers, such as [7], [8], and [9]. This enables the defense side to fine tune their HT detectors to showcase strong HT detection rates. We believe that a standard benchmark should contain several HT-infected and HT-free instances to provide better dataset balance for researchers. Fig. 1 shows the difference between existing benchmark approaches and our proposed method.

In this letter, we assume that the attacker relies on the third-party electronic design automation (3P-EDA) tools to insert HTs at the synthesis stage [10]. The EDA tool enables the attacker to restructure the circuit's netlist while maintaining the circuit functionality unchanged. It is worth mentioning that functional restructuring used in this study is inherently different from hardware obfuscation in the logic locking context [11]. We utilize ABC [12], an open-source logic optimization tool, leveraging its various representations and optimization techniques to modify the structure of circuits. Our released benchmark will be called *Seeker1*, and it will be publicly available for researchers to test their tools [13]. Our methodology is described so that we can employ this benchmark creation approach to more realistic circuits, such as microcontrollers and microprocessors.

The contributions of this letter are as follows.

1) We introduce The Seeker's Dilemma, a formal definition of HT detection similar to finding HTs in real-world scenarios.
2) We introduce *Seeker1* (a benchmark with potentially hidden HTs), and the methodology we use to create this benchmark to help improve HT detection.
3) We use principal component analysis (PCA) to analyze the significance of *Seeker1* from a machine learning (ML) training perspective.

II. Related Work

HT detection methods can be categorized based on their dependence on a golden model. If security engineers have access to a golden model or its specification, they will look for any deviations in the functionality of the devices from the expected standard behavior when applying test vectors. Previous work, such as [3], [4], [6], and [14] employ analytical and ML-based approaches to generate test vectors that they

Manuscript received 6 August 2024; accepted 9 August 2024. This work supported in part by the NSF under Grant 2219680 and Grant 2219679. This manuscript was recommended for publication by A. Shrivastava. *(Corresponding author: Amin Sarihi.)*

Amin Sarihi and Abdel-Hameed A. Badawy are with the Klipsch School of Electrical and Computer Engineering, New Mexico State University, Las Cruces, NM 88003 USA (e-mail: sarihi@nmsu.edu).

Ahmad Patooghy is with the Department of Computer Systems Technology, North Carolina Agricultural and Technical State University, Greensboro, NC 27411 USA.

Peter Jamieson is with the Department of Electrical and Computer Engineering, Miami University, Oxford, OH 45056 USA.

Digital Object Identifier 10.1109/LES.2024.3443155

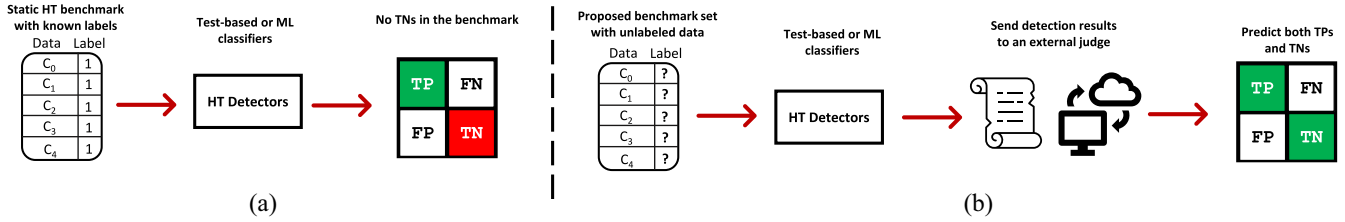


Fig. 1. Comparison of (a) current HT detection approaches with static HT benchmarks that only contain HT-infected circuits versus (b) our proposed HT detection flow, including restructured benchmarks with and without HTs.

believe would best expose the existence of malicious HTs. When interpreting HT classification results, we encounter four possible outcomes: 1) false positive (FP); 2) false negative (FN); 3) true positive (TP); and 4) true negative (TN). Test-based HT detection would never lead a security engineer to an FP case. In addition to using test vectors, other ML-based approaches have also been used extensively toward HT detection when the golden models are not available [2], [5], [15]. The accuracy of such approaches solely rely on the data on which they were trained (which is the Trusthub [7] benchmark in most cases). These detectors can result in all four classification outcomes.

HT benchmarks have also been the subject of study by various researchers. Trusthub [1], [16] was among the first ones where several HT designs are available to study. Despite its valuable contribution to the HT research community, the benchmark lacks the size and variety needed to push the detection field forward [8]. Other researchers have made efforts to address Trusthub's shortcomings by developing automated tools that generate HT benchmarks [8], [9], [14], [17]. While pushing the HT field forward, the newly introduced benchmarks are heavily unbalanced where the number of HT-infected data points outnumber the HT-free ones, which leads to training biased ML-based HT detectors. Our work strives to cover the deficiencies of the previous literature by introducing *Seeker1*. The details of our benchmark will be explained in the remainder of this letter.

III. Seeker's Dilemma

The *Hide&Seek* problem as related to cybersecurity [18] in the HT domain (a situation which we call The Seeker's Dilemma) is that given an IC or netlist, the *Seeker* (S), does not know whether an HT has been hidden in the netlist in a two-player game, where H is the *Hider* and S is the *Seeker* (detector). We define $k = |\mathcal{H}|$ as the number of objects hidden, and in The Seeker's Dilemma, k hidden objects have the condition $k \geq 0$. Moreover, the value of k is unknown by the *Seeker*. Adding this condition and the unknown information transforms the problem into what we define as The Seeker's Dilemma, and from a complexity theory perspective, makes the problem significantly harder from a real-world perspective. The Seeker's Dilemma has very different strategical implications such that the *Hider* hides k objects (could be nothing if $k = 0$) and the *Seeker* (S) searches for hidden objects in L queries or moves, where H tries to maximize L and, conversely, S, tries to minimize L.

The major challenge with existing HT benchmarks is that these circuits are known-knowns in terms of the existence of HTs, i.e., almost in all cases, $k = 1$. This means that both the inserting and detecting sides know the situation, which we believe oversimplifies the problem. Despite being very helpful to the community, Trusthub benchmarks [7] fall into this class. To address this gap, we have created The Seeker's Dilemma HT benchmark (we call it *Seeker1*) featuring $k \geq 0$. The *Seeker1* benchmark suite will be publicly available.

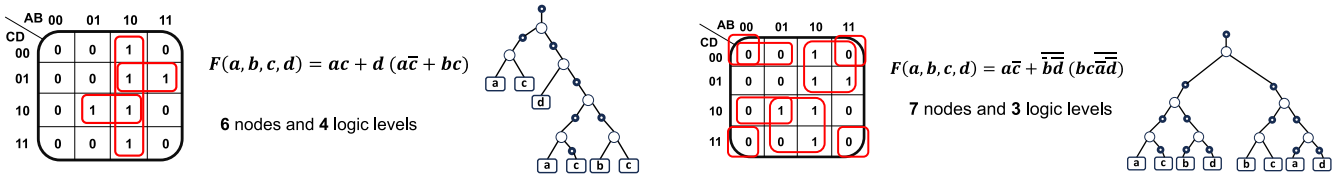
A. Benchmark Generation

We have selected the original circuits for *Seeker1* from ISCAS-85's [19] combinational designs that have been used throughout CAD research community [4], [5], [8], [9]. This benchmark has been widely used to evaluate the effectiveness of logic synthesis tools, technology mapping algorithms, test generation algorithms, timing analysis, various optimization techniques for digital circuits, and HT detection. So, we believe that ISCAS-85 is the first candidate to generate *Seeker1*.

The AND-inverter graph (AIG) format is used to facilitate the functional restructuring of our target circuits. AIG represents circuits with two-input *AND* gates and *NOT* gates [20], derived using DeMorgan's rule. AIGs, though noncanonical, are used in algorithms to optimize area, delay, and formal equivalence checking [21]. An example is shown in Fig. 2. We use ABC [12] and employ 18 functional restructuring methods to alter ISCAS-85 circuits, hiding HTs and complicating detection. Our benchmark includes 8 ISCAS-85 circuits ($c880$, $c1355$, $c1908$, $c2670$, $c3540$, $c5315$, $c6288$, and $c7552$) with variably added HTs, details of which are withheld to prevent reverse-engineering. We use the HT circuits generated by an RL framework explained in [14] and [17]. We pick 100 inserted HTs from each circuit and create 18 versions of each. We also use ABC to functionally restructure HT-free circuits to add further complexity to *Seeker1*.

IV. Analysis of Our Benchmark

To analyze our first Seeker's Dilemma benchmark, *Seeker1*, we test the benchmark against the existing HT detection tools. We use three different HT detection strategies/tools to measure the quality of inserted HTs: 1) the test vectors from RL_HT_DETECT developed in [6] and [14]; 2) the test vectors from DETERRENT proposed in [4]; and 3) the open-source HW2VEC [5]. RL_HT_DETECT and DETERRENT are test-based HT detectors requiring a golden model, which is resilient against functional restructuring techniques as these do not alter circuit functionality. HW2VEC, a GNN-based HT detector, extracts behavioral features from hardware designs to train a binary classifier on a dataset with 200 features, with its performance depending on the dataset quality and quantity. Next, we train a binary classifier with the following training data.



(a) (b)

Fig. 2. Two representations of a circuit with the same truth table. Representation #1 is aimed at improving the area in terms of comparatively fewer nodes, while representation #2 enhances delay with fewer logic levels. (a) AIG representation #1. (b) AIG representation #2.

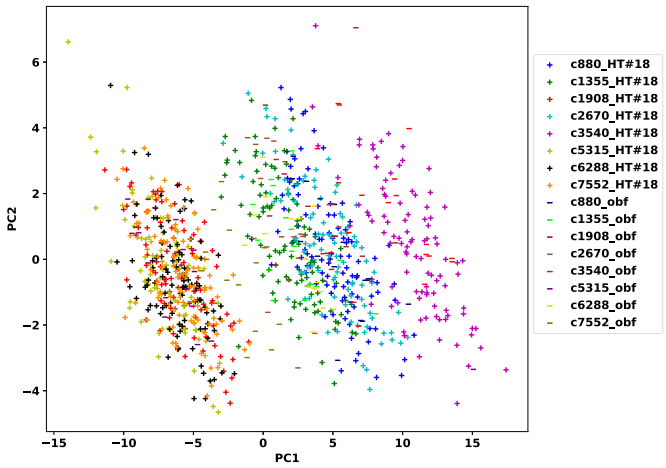


Fig. 3. PCA analysis of hidden HTs versus clean functionally transformed ISCAS-85 circuits.

1) The *TJ_RTL* dataset used in [5] for training HW2VEC. This dataset contains communication protocols and encryption algorithms from Trusthub. The dataset contains 26 HT-infected and 11 HT-free instances. The remainder of this letter refers to the HT detection reports under this scenario as S_1.
2) We add two versions of ISCAS-85 HT-free benchmarks [8], [19] to *TJ_RTL* to make the dataset labels more balanced. This action adds 16 more HT-free instances to the previous set. We refer to the HT detection reports under this scenario as S_2 hereafter.

Fig. 3 shows the outcome of the PCA analysis [22] for the functionally restructured versions of both infected (ending with _HT#18 suffix) and clean (ending with _obf suffix) ISCAS-85 circuits with restructuring technique number 18. We transform each circuit with HW2VEC and extract 200 attributes for each one. PCA is used to reduce dimensionality to a handful of principal components that collectively explain a significant portion of the total variance. As can be seen, the HT-free instances (marked as -) are distributed among the HT-infected circuits (marked as +) when we plot PC1 against PC2. It is hard to find a boundary upon which these two classes can be separated. To further study this, we train HW2VEC with S_1 and S_2, and investigate the classification accuracy of HT-free data points. Among the ISCAS-85 benchmark circuits, the restructured HT-free instances of $c5315$, $c6288$, and $c7552$ have the highest *FP* rates of 80%, 40%, and 45% on average, while the respective figures for the remaining circuits are under 20%. This experiment emphasizes the need for a diversified dataset for training HT detectors.

Fig. 4 shows the heatmap of HT detection accuracy percentages (*TPs*) for the functionally restructured HT-infected circuits using HW2VEC trained with both S_1 and S_2. Each circuit-functional-restructuring method pair contains 100 HT-infected circuits generated by the RL-inserter and functionally equivalent transformations using ABC. The detection accuracy in Fig. 4(a) ranges between 0% and 80% for S_1 while the same figure ranges between 0% and 20% in Fig. 4(b) for S_2. In both detection scenarios, the circuits are divided into two groups: 1) {$c880$, $c1355$, $c2670$, $c3540$} and 2) {$c1908$, $c5315$, $c6288$, $c7552$}.

While HW2VEC detects up to 80% of HTs in the second group under $S1$, it significantly underperforms with the first group. The situation worsens under $S2$, where the detector fails to classify HT-infected circuits in group "1" while the figures are only slightly better for group "2". The underlying reason can be sought with the mixture of labels in each S_1 and S_2 and the unseen hidden data. The extra HT-free labels in S_2 bias the detector to classify more instances as HT-free.

We also compare *Seeker1* with two existing benchmarks introduced in [8] and [9]. Both benchmarks only contain HT-infected circuits and HTs are inserted using low signal switching nets. For [8], we train HW2VEC under S_1 and we report the detection accuracies for four reported ISCAS-85 benchmarks, *c2670, c3540, c5315,* and *c6288*. The detection figures are 100%, 0%, 70%, and 0%, respectively. In [9], there are two ISCAS-85 benchmarks: 1) $c6288$ and 2) $c7552$. The detection rates are 10% and 90%, respectively. Compared with Fig. 4(a), *Seeker1* evades detection more consistently throughout the entire benchmark. It is important to note that the insertion criteria of [14] and [17] are inherently different from that of [8] and [9]. In the future, we plan to investigate the impact of various HT insertion and functional restructuring strategies on HT detectors.

Fig. 5 shows the detection percentage for RL_HT_DETECT (*Combined*), DETERRENT, and HW2VEC for the original HT-infected circuits (ending with _RL suffix) and their ABC-functionally equivalent transformed versions (ending with _ABC suffix). The x axis shows the benchmark circuits and the y axis shows the HT detection accuracy (*TPs*) as a percentage. To fairly compare against DETERRENT, we only mention the four ISCAS-85 benchmarks studied in the DETERRENT paper. As can be seen, the detection accuracy of RL_HT_DETECT is higher than DETERRENT in all four circuits; however, the difference is more substantial in $c2670$ and $c5315$. The reason can be sought into multicriteria [6] versus single criterion [4] HT detection. As for HW2VEC's detection rate under S_1 and S_2 for the baseline RL benchmarks is nearly 100%. The situation differs for the functionally equivalent transformed HTs, with lower HT detection rates.

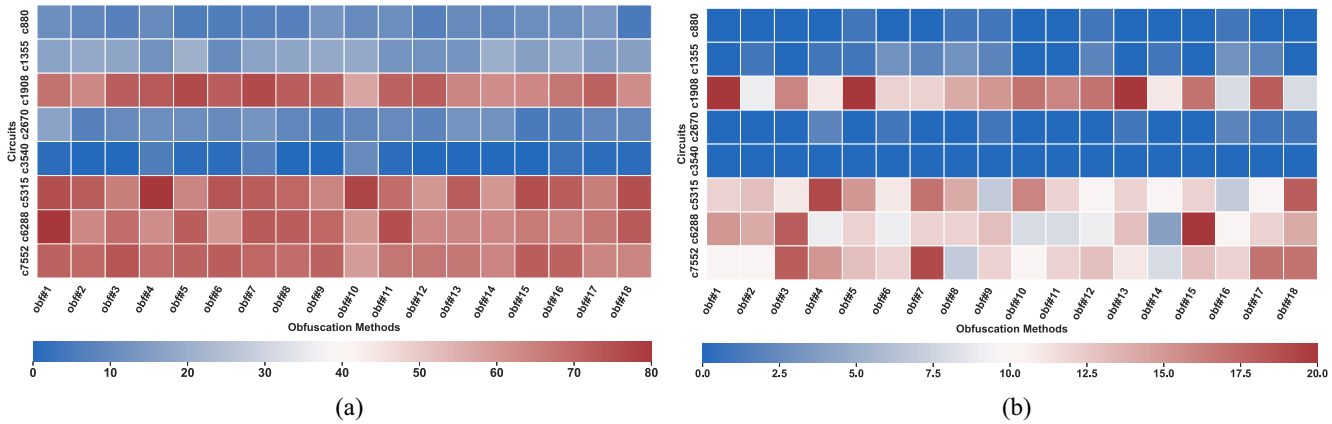


Fig. 4. Detection accuracy of 18 functionally equivalent transformation methods trained with (a) S_1 and (b) S_2 for ISCAS-85.

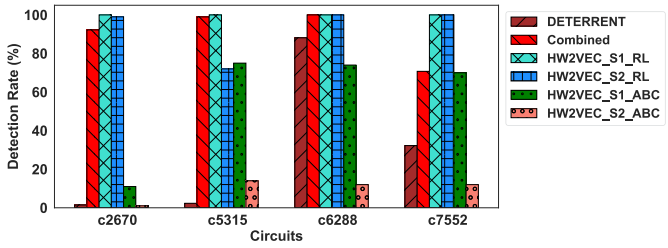


Fig. 5. Detection rate (TPs) of DETERRENT [4], RL_HT_DETECT [14], and HW2VEC [5] for hidden HT-infected ISCAS-85 circuits.

V. Conclusion

This letter defines HT detection as a *Hide&Seek* game termed The Seeker's Dilemma, highlighting the challenge of detecting infected circuits without prior knowledge. Using this paradigm, a new benchmarking strategy is proposed for evaluating HT detection methods more accurately. The innovative problem statement and strategy aim to foster new ideas and improve quality assessments in HT detection. A combinational benchmark was created and released to guide future work in HT detection and insertion. Existing HT detection methods were tested on this benchmark, marking the first such evaluation. Future plans include training a more robust HT detector against data variations.

References

[1] B. Shakya, T. He, H. Salmani, D. Forte, S. Bhunia, and M. Tehranipoor, "Benchmarking of hardware Trojans and maliciously affected circuits," *J. Hardw. Syst. Security*, vol. 1, no. 1, pp. 85–102, 2017.

[2] K. Hasegawa, M. Yanagisawa, and N. Togawa, "Trojan-feature extraction at gate-level netlists and its application to hardware-trojan detection using random forest classifier," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, 2017, pp. 1–4.

[3] Z. Pan and P. Mishra, "Automated test generation for hardware trojan detection using reinforcement learning," in *Proc. 26th Asia South Pacific Design Autom. Conf.*, 2021, pp. 408–413.

[4] V. Gohil, S. Patnaik, H. Guo, D. Kalathil, and J. Rajendran, "DETERRENT: Detecting trojans using reinforcement learning," in *Proc. 59th ACM/IEEE Design Autom. Conf.*, 2022, pp. 697–702.

[5] S.-Y. Yu, R. Yasaei, Q. Zhou, T. Nguyen, and M. A. Al Faruque, "HW2VEC: A graph learning tool for automating hardware security," in *Proc. IEEE Int. Symp. Hardw. Oriented Security Trust (HOST)*, 2021, pp. 13–23.

[6] A. Sarihi, P. Jamieson, A. Patooghy, and A.-H. A. Badawy, "Multi-criteria hardware Trojan detection: A reinforcement learning approach," in *Proc. IEEE 66th Int. Midwest Symp. Circuits Syst. (MWSCAS)*, 2023, pp. 1093–1097.

[7] "Trust-hub." Accessed: Nov. 8, 2023. [Online]. Available: https://trust-hub.org/

[8] J. Cruz, Y. Huang, P. Mishra, and S. Bhunia, "An automated configurable trojan insertion framework for dynamic trust benchmarks," in *Proc. Design, Autom. Test Eur. Conf. Exhibit. (DATE)*, 2018, pp. 1598–1603.

[9] V. Gohil, H. Guo, S. Patnaik, and J. Rajendran, "ATTRITION: Attacking static hardware trojan detection techniques using reinforcement learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2022, pp. 1275–1289.

[10] 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.

[11] T. Hoque, R. S. Chakraborty, and S. Bhunia, "Hardware obfuscation and logic locking: A tutorial introduction," *IEEE Design Test*, vol. 37, no. 3, pp. 59–77, Jun. 2020.

[12] R. Brayton and A. Mishchenko, "ABC: An academic industrial-strength verification tool," in *Proc. 22nd Int. Conf., Comput. Aided Verif.*, 2010, pp. 24–40.

[13] "Seeker's dilemma hardware trojan-benchmarks: Functionally restructured hardware trojan benchmarks." Accessed: Feb. 27, 2024. [Online]. Available: https://github.com/NMSU-PEARL/Seeker-s-Dilemma-Hardware-Trojan-Benchmarks

[14] A. Sarihi, A. Patooghy, P. Jamieson, and A.-H. A. Badawy, "Trojan playground: A reinforcement learning framework for hardware trojan insertion and detection," *J. Supercomput.*, vol. 80, pp. 14295–14329, Jul. 2024.

[15] H. Salmani, "COTD: Reference-free hardware trojan detection and recovery based on controllability and observability in gate-level netlist," *IEEE Trans. Inf. Forensics Security*, vol. 12, pp. 338–350, 2016.

[16] H. Salmani, M. Tehranipoor, and R. Karri, "On design vulnerability analysis and trust benchmarks development," in *Proc. IEEE 31st Int. Conf. Comput. Design (ICCD)*, 2013, pp. 471–474.

[17] A. Sarihi, A. Patooghy, P. Jamieson, and A.-H. A. Badawy, "Hardware Trojan insertion using reinforcement learning," in *Proc. Great Lakes Symp. VLSI*, 2022, pp. 139–142.

[18] M. Chapman, G. Tyson, P. McBurney, M. Luck, and S. Parsons, "Playing hide-and-seek: An abstract game for cyber security," in *Proc. 1st Int. Workshop Agents CyberSecurity*, 2014, pp. 1–8.

[19] M. C. Hansen, H. Yalcin, and J. P. Hayes, "Unveiling the ISCAS-85 benchmarks: A case study in reverse engineering," *IEEE Design Test Comput.*, vol. 16, no. 3, pp. 72–80, Jul.–Sep. 1999.

[20] A. Mishchenko and R. Brayton, "Integrating an AIG package, simulator, and SAT solver," in *Proc. Int. Workshop Logic Synth. (IWLS)*, 2018, pp. 11–16.

[21] A. B. Chowdhury, B. Tan, R. Karri, and S. Garg, "OpenABC-D: A large-scale dataset for machine learning guided integrated circuit synthesis," 2021, *arXiv:2110.11292*.

[22] R. Bro and A. K. Smilde, "Principal component analysis," *Anal. Methods*, vol. 6, no. 9, pp. 2812–2831, 2014.