

CSAW AI Hardware Attack Challenge Round 1 Report

Shubhi Shukla, Upasana Mandal, Tishya Sarma Sarkar and Kislay Arya
Secured Embedded Architecture Laboratory (SEAL)
Indian Institute of Technology Kharagpur

October 18, 2024

Contents

1	Introduction	2
2	Proposed Methodology for Automated LLM tool	4
3	Preliminary Findings/Results	6
4	Conclusion	9

Chapter 1

Introduction

In an era where modern electronics power critical infrastructure and everyday devices, the trustworthiness of hardware has become a significant concern. The increasing adoption of third-party intellectual property (3PIP) blocks, commercial off-the-shelf (COTS) components, and electronic design automation (EDA) tools has introduced untrusted elements into the hardware supply chain. This shift has amplified the risk of malicious hardware alterations, particularly through hardware Trojans—stealthy, malicious modifications that can severely compromise the security and functionality of digital systems.

Hardware Trojans represent a unique and dangerous threat. These malicious alterations can be designed to leak sensitive information, disrupt system performance, manipulate data integrity, create hidden backdoors for unauthorized persistent access, or enable privilege escalation, granting attackers full control of the system. Their stealthy nature allows them to evade traditional verification methods, making detection an extremely challenging task for security engineers. The expanding complexity of hardware designs only further complicates the task, as the insertion of Trojans can occur at multiple stages of the design and manufacturing process, often involving third-party entities.

There have been several efforts aimed at detecting vulnerabilities in hardware designs, focusing on uncovering potential threats like Trojan insertions[KEZ⁺24]. However, the sheer diversity of Trojan types and the techniques for inserting them make it exceedingly difficult to comprehensively check each design for all possible Trojans. Trojans can be inserted through various methods, such as manipulating the critical path to degrade system performance, altering signal probabilities to activate under specific conditions, or hiding within unused circuit paths that bypass standard verification. Other techniques include introducing Trojans via power consumption variations that evade detection through abnormal power signatures, inserting malicious logic into clocking mechanisms to cause unpredictable behavior, or leveraging side-channel vulnerabilities that operate covertly, leaving minimal traces in the normal operation of the system [XFJ⁺16]. Given the vast range of Trojan types and insertion techniques, traditional approaches face significant limitations in providing comprehensive coverage. As hardware designs grow in complexity, manually verifying each potential vulnerability becomes increasingly impractical. The variety of attack vectors, stealthy behavior, and activation mechanisms makes it challenging to ensure robust security against all possible Trojan threats.

This challenge motivates the need for a more advanced solution, leading us to propose an automated tool that leverages large language models (LLMs) for hardware security. LLMs offer significant advantages because they are trained on vast datasets, including knowledge about hardware Trojans, vulnerabilities, and attack techniques. Using this foundational knowledge, our tool intelligently evaluates the digital design’s netlist, creates a Directed Acyclic Graph (DAG) of the design, and ranks the probability of occurrences in the nets. Additionally, LLMs possess the capability to understand and generate Hardware Description Languages (HDLs) such as Verilog, enabling them to analyze design code at a low level and suggest vulnerable nets where a Trojan could be inserted. By combining these properties, our LLM-based automated tool not only identifies potential vulnerabilities but also generates trigger and payload models for Trojans, automating the entire process from analysis to the generation of exploitable Trojan logic.

Existing methods, such as Taint[JKKK17], HAL[FWS⁺19], TRIT[CHMB18], MIMIC[CGN⁺22] and other works[GSL⁺23] offer useful tools for detecting hardware Trojans, but they come with significant drawbacks. They often rely on manual effort, where users must carefully select where to insert Trojans or focus on narrow criteria, limiting the scope of threats they can identify. These

approaches are time-consuming, difficult to scale, and often miss more complex or varied types of Trojans. In contrast, our LLM-based method offers a more advanced and comprehensive solution. By leveraging the power of large language models, which already have knowledge about different types of Trojans and hardware vulnerabilities, our tool can automatically identify weaknesses and suggest Trojan insertion points without requiring manual input. This not only saves time but also allows for a deeper and broader exploration of potential security threats, making our method more scalable and effective in protecting modern hardware designs.

Chapter 2

Proposed Methodology for Automated LLM tool

In this section, we discuss the overall workflow of our proposed tool for AI-based hardware attacks. We leverage GPT-based LLMs to identify the vulnerable nets in the input digital design by evaluating each net's probability of occurrence and ranking them accordingly. The LLM model creates a Directed Acyclic Graph (DAG) to rank the probability of occurrences. This process further generates candidate trigger and payload models that assist in determining the appropriate variant of hardware Trojans. Following the trigger and payload model generation, a set of valid and suitable hardware Trojan designs are created and incorporated into the original circuit to validate the strength of the generated Trojan. Finally, this Trojan is exploited to reveal any secret information of the circuit or impart fault in the design. This methodology is summarized in Fig.2.1.

Digital Design Netlist Feed to LLM for DAG Generation:

The process begins with the input of the digital design's netlist into a Large Language Model (LLM) that subsequently generates a Directed Acyclic Graph (DAG). In this context, the netlist—a detailed representation of the circuit design, including the components and inter-connections—undergoes parsing by the LLM. The DAG serves as a structural representation of the design's data flow, capturing the dependencies and hierarchy of operations. The DAG creation enables a granular, computationally tractable view of the design, which will facilitate the subsequent steps in Trojan modeling and insertion.

Trigger and Payload Model Generation:

Following the DAG creation, models for both the trigger and payload of a potential hardware Trojan are generated. The trigger model simulates the specific conditions under which the Trojan would be activated within the digital design. These conditions might include specific input

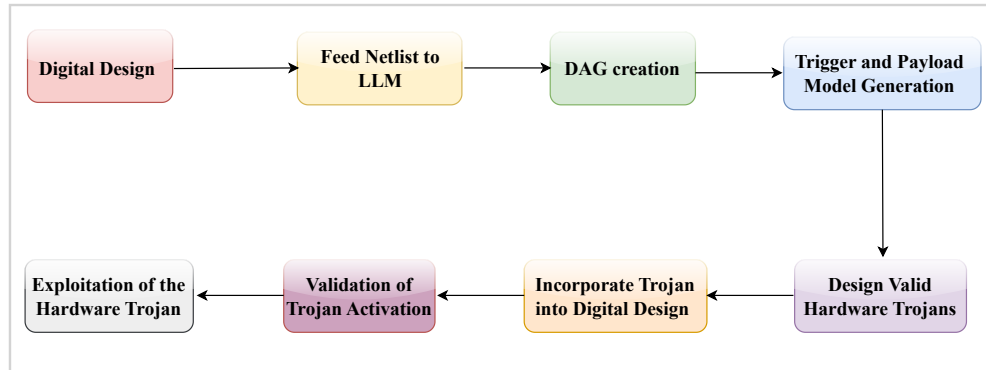


Figure 2.1: Tool Program Flow

sequences, timing events, or other unusual activity patterns. Meanwhile, the payload model defines the actions executed once the Trojan is triggered, such as data leakage, performance degradation, or system sabotage. This phase is critical for ensuring that the Trojan's behavior is both feasible within the design and targeted toward specific vulnerabilities.

Design Validation of Hardware Trojans:

The next step is the validation of the hardware Trojan design. The term "validation" refers to verifying that the designed Trojan is compatible with the existing digital architecture and functions as expected within the design constraints. This stage involves ensuring that the Trojan can be incorporated without being immediately flagged by standard design verification tools. Techniques such as formal verification, simulation, and synthesis may be employed to check that the Trojan operates stealthily within normal design processes, awaiting activation based on the modeled triggers.

Trojan Incorporation into Digital Design

Upon successful validation, the Trojan is incorporated into the original digital design. This step involves the modification of the circuit's netlist or logic to embed the Trojan. Careful consideration is given to the Trojan's placement to minimize detectability during typical design verification procedures. The objective is to ensure that the Trojan remains dormant under normal operating conditions and is only activated under the specific scenarios defined in the trigger model. This requires intricate knowledge of the design flow and a strategic insertion process to ensure operational stealth.

Trojan Activation Validation:

The final phase of the workflow is the validation of Trojan activation. This step involves simulating or testing the modified digital design to verify that the Trojan activates under the defined conditions. Tools such as hardware emulation, logic simulation, or formal analysis may be used to confirm that the Trojan's trigger operates correctly and that the payload is executed as intended. This validation ensures that the Trojan not only integrates with the design but also performs its malicious function as expected, without disrupting the design's regular functionality unless triggered.

Chapter 3

Preliminary Findings/Results

We validate the efficiency of our proposed AI-based tool on the c17 benchmark circuit from the IS-CAS'85 digital design series. We selected this dataset for a preliminary evaluation as the ISCAS'85 is a dataset consisting of purely combinational circuits, thus reducing the complexities of the sequential counterparts. The following chatbox provides an overview of the proposed attack mounted by the tool, wherein it first converts the provided circuit into a DAG to determine the probable trigger nets. The tool identifies an unusual combination of inputs, causing the output signal N23 to suddenly go high. This imparts a fault in the original functionality of the circuit. Thus the corresponding malicious alteration of the Verilog code is considered as a hardware Trojan. The corresponding simulation results are displayed in Fig.3.1 and Fig.3.2 in which the changes in the states of N23 are analyzed and compared without and with the Trojan-infected circuit, respectively. A minor change of state of the signal N23 is visible in the waveforms. This enables us to claim that our proposed AI-based tool has the capability of mounting a hardware Trojan in a vulnerable circuit and exploiting that Trojan to alter the expected functionality of a design.

GPT Capabilities

User: Depending on the parameters of the graph please suggest whether a hardware trojan can be inserted inside the verilog code provided.

GPT:

Critical Paths: Trojans are often inserted into non-critical paths to avoid detection and minimize performance impact. If a graph has long critical paths or nodes with high connectivity, it might indicate areas where additional logic can be stealthily integrated.

Redundant Paths: Nodes or subgraphs that perform redundant computations or have similar operations might provide opportunities to insert Trojans. For example, if there are multiple similar paths in the mixcolumn or shiftrow modules, these could be potential sites for Trojan insertion.

Idle or Unused Components: Any logic components that aren't always active or are only sporadically engaged can hide Trojan circuitry. These could be nodes with fewer incoming or outgoing edges. Analyzing connectivity can reveal if a component is used irregularly, as this would make it a good candidate for embedding malicious code.

Our Result leveraging LLM

1. DAG Adjacency List

N1: N10
N2: N16
N3: N10, N11
N6: N11
N7: N19
N10: N22
N11: N16, N19
N16: N22, N23
N19: N23
N22:
N23:

2. Trojan Insertion and Verilog Code

Yes, a hardware trojan can be inserted. One possible approach is to make output N23 go high conditionally based on a rare combination of inputs.

Verilog

=====START=====

```
module c17.trojan (N1, N2, N3, N6, N7, N22, N23);
input N1, N2, N3, N6, N7;
output N22, N23;
wire N10, N11, N16, N19;
wire trojan_active;
nand NAND2_1 (N10, N1, N3);
nand NAND2_2 (N11, N3, N6);
nand NAND2_3 (N16, N2, N11);
nand NAND2_4 (N19, N11, N7);
nand NAND2_5 (N22_normal, N10, N16);
nand NAND2_6 (N23_normal, N16, N19);
// Trojan condition: if all inputs are high (N1 = N2 = N3 = N6 = N7 = 1), activate Trojan assign tro-
jan_active = N1 N2 N3 N6 N7;
// Normal outputs when Trojan is inactive
assign N22 = (trojan_active) ? 1'b0 : N22_normal;
assign N23 = (trojan_active) ? 1'b1 : N23_normal;
endmodule
=====END=====
```

3. Trojan Explanation and Trigger

The trojan added forces the output N23 high if the specific condition N1 N2 N3 N6 N7 is met. Otherwise, the circuit behaves as expected, generating its normal output.

The trojan condition is designed to trigger when all inputs (N1, N2, N3, N6, and N7) are high. This combination might be rare in typical circuit operation, making it hard to detect during normal testing. Once the rare combination occurs, N23 is forced high, which could cause unintended behavior in the larger system. Depending on the function of N23, this could lead to system malfunctions, data corruption, or even failure.

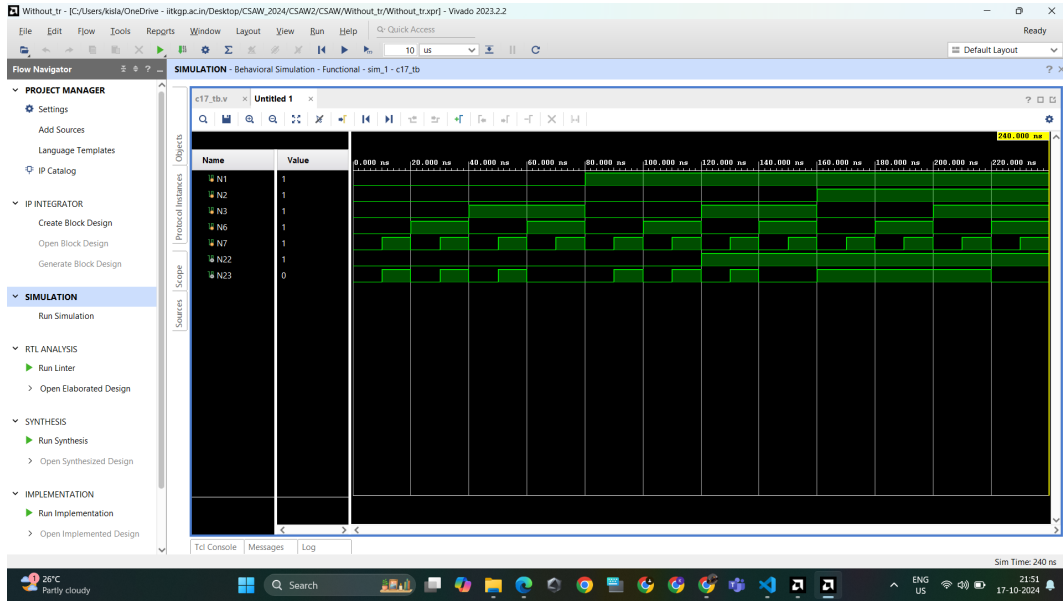


Figure 3.1: Simulation results of c17 benchmark dataset from ISCAS'85 without hardware Trojan

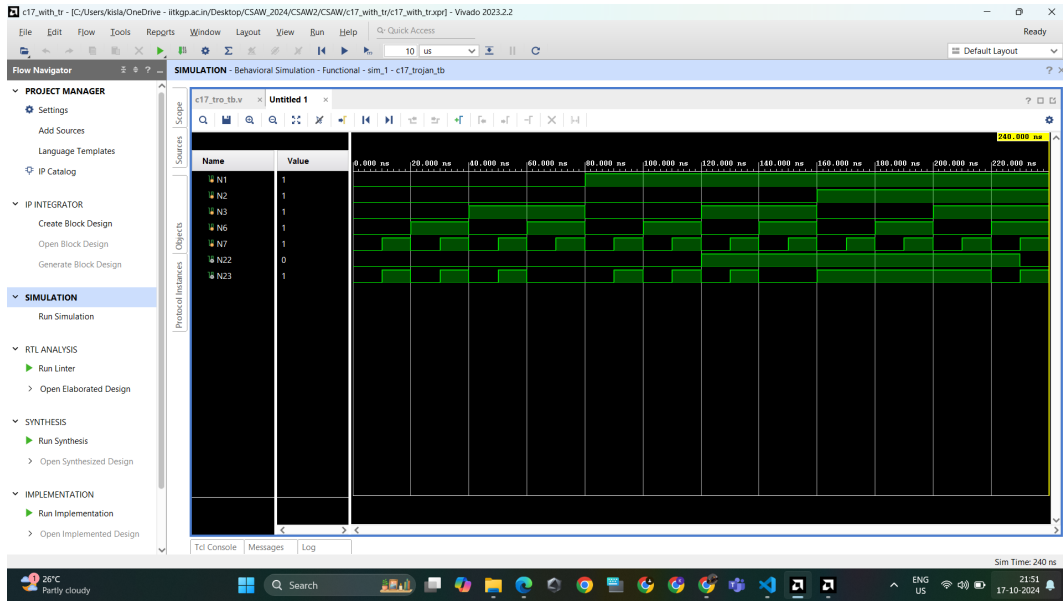


Figure 3.2: Simulation results of c17 benchmark dataset from ISCAS'85 with hardware Trojan

Chapter 4

Conclusion

In this challenge, we propose an AI-driven automated tool to insert hardware Trojans in open-source digital designs by exploiting the corresponding HDL or netlist. This AI-based hardware attack is an invasive procedure that leverages the power of GPT-based LLMs to detect candidate trigger and payload nets in the design. The trigger and payload models are ranked based on their probability of occurrence. We use the c17 benchmark dataset from ISCAS'85 to validate the efficacy of our proposed tool. From our preliminary experiments, we can conclude that our tool successfully detects the vulnerable nets in the dataset, designs a valid hardware Trojan corresponding to the chosen net, and mounts this Trojan into the original circuit. This allows us to exploit the functionality of the design by imparting an abrupt state change of an output signal, which is equivalent to a fault attack. Furthermore, we aim to validate the proposed tool on an open-source Advanced Encryption System (AES) design and other benchmark datasets.

Bibliography

- [CGN⁺22] Jonathan Cruz, Pravin Gaikwad, Abhishek Nair, Prabuddha Chakraborty, and Swarup Bhunia. Automatic hardware trojan insertion using machine learning. *CoRR*, abs/2204.08580, 2022.
- [CHMB18] Jonathan Cruz, Yuanwen Huang, Prabhat Mishra, and Swarup Bhunia. An automated configurable trojan insertion framework for dynamic trust benchmarks. In Jan Madsen and Ayse K. Coskun, editors, *2018 Design, Automation & Test in Europe Conference & Exhibition, DATE 2018, Dresden, Germany, March 19-23, 2018*, pages 1598–1603. IEEE, 2018.
- [FWS⁺19] Marc Fyrbiak, Sebastian Wallat, Pawel Swierczynski, Max Hoffmann, Sebastian Hoppach, Matthias Wilhelm, Tobias Weidlich, Russell Tessier, and Christof Paar. HAL - the missing piece of the puzzle for hardware reverse engineering, trojan detection and insertion. *IEEE Trans. Dependable Secur. Comput.*, 16(3):498–510, 2019.
- [GSL⁺23] Kevin Immanuel Gubbi, Banafsheh Saber Latibari, Anirudh Srikanth, Tyler Sheaves, Sayed Arash Beheshti-Shirazi, Sai Manoj PD, Satareh Rafatirad, Avesta Sasan, Houman Homayoun, and Soheil Salehi. Hardware trojan detection using machine learning: A tutorial. *ACM Trans. Embed. Comput. Syst.*, 22(3), April 2023.
- [JKKK17] Vinayaka Jyothi, Prashanth Krishnamurthy, Farshad Khorrami, and Ramesh Karri. Taint: tool for automated insertion of trojans. In *2017 IEEE International Conference on Computer Design, ICCD 2017, Boston, MA, USA, November 5-8, 2017*, pages 545–548. IEEE Computer Society, 2017.
- [KEZ⁺24] Johann Knechtel, Mohammad Eslami, Peng Zou, Min Wei, Xingyu Tong, Binggang Qiu, Zhijie Cai, Guohao Chen, Benchao Zhu, Jiawei Li, Jun Yu, Jianli Chen, Chun-Wei Chiu, Min-Feng Hsieh, Chia-Hsiu Ou, Ting-Chi Wang, Bangqi Fu, Qijing Wang, Yang Sun, Qin Luo, Anthony W. H. Lau, Fangzhou Wang, Evangeline F. Y. Young, Shunyang Bi, Guangxin Guo, Haonan Wu, Zhengguang Tang, Hailong You, Cong Li, Ramesh Karri, Ozgur Sinanoglu, and Samuel Pagliarini. Trojan insertion versus layout defenses for modern ICs: Red-versus-blue teaming in a competitive community effort. Cryptology ePrint Archive, Paper 2024/1440, 2024.
- [XFJ⁺16] Kan Xiao, Domenic Forte, Yier Jin, Ramesh Karri, Swarup Bhunia, and Mark M. Tehranipoor. Hardware trojans: Lessons learned after one decade of research. *ACM Trans. Design Autom. Electr. Syst.*, 22(1):6:1–6:23, 2016.