SEAL THE TROJAN

Shubhi Shukla [1]   Tishya Sharma Sarkar [1]   Upasana Mandal [1]   Kislay Arya

[1]Indian Institute of Technology Kharagpur

shuklashubhi6@gmail.com   tishyasarmasarkar@gmail.com   mandal.up98@kgpian.iitkgp.ac.in   kislayarya536@gmail.com

Introduction

- Hardware Trojans introduce significant security risks by enabling stealthy manipulation, data leakage, and privilege escalation within systems, making detection a critical challenge due to their hidden and complex nature.
- Traditional Trojan detection tools, like TAINT, HAL, and MIMIC, require manual input and are often narrowly focused, struggling to keep up with diverse and evolving Trojan insertion techniques, which limits their effectiveness and scalability.
- By leveraging large language models (LLMs) like Google's Gemini and OpenAI's GPT-4, our tool automatically analyzes digital designs, generating Directed Acyclic Graphs (DAGs) to evaluate and rank net probabilities, enhancing Trojan detection accuracy across complex designs.
- Our LLM-based tool can generate Verilog code with embedded Trojan models, automating the creation of both trigger and payload models, thus streamlining vulnerability assessment and Trojan insertion processes.
- The tool's ability to interpret and generate Hardware Description Languages (HDLs), including Verilog, allows it to detect weak points in design code and provide targeted vulnerability insights, significantly advancing hardware security efforts.

Proposed Methodology for Automated LLM tool

- Utilizes large language models (LLMs) to automate the identification and insertion of hardware trojans into Verilog code, leveraging graph-based code analysis and LLM-guided insertion for a systematic approach to hardware security testing.
- Analyzes Verilog code by creating a directed acyclic graph (DAG) to highlight logical flow and structural dependencies, enabling the LLM to identify exploitable locations with minimal impact on functionality and maximum vulnerability potential.
- Employs LLM1 to insert hardware trojans based on specific insertion methods, such as FSM targeting, operator-based, and bit flip techniques, ensuring trojan placement aligns with functional and logical requirements.
- Uses a multi-step verification process with three LLMs (LLM1, LLM2, and LLM3) to iteratively refine and validate the trojan insertion, ensuring it meets desired specifications and adheres to the original Verilog structure without errors.

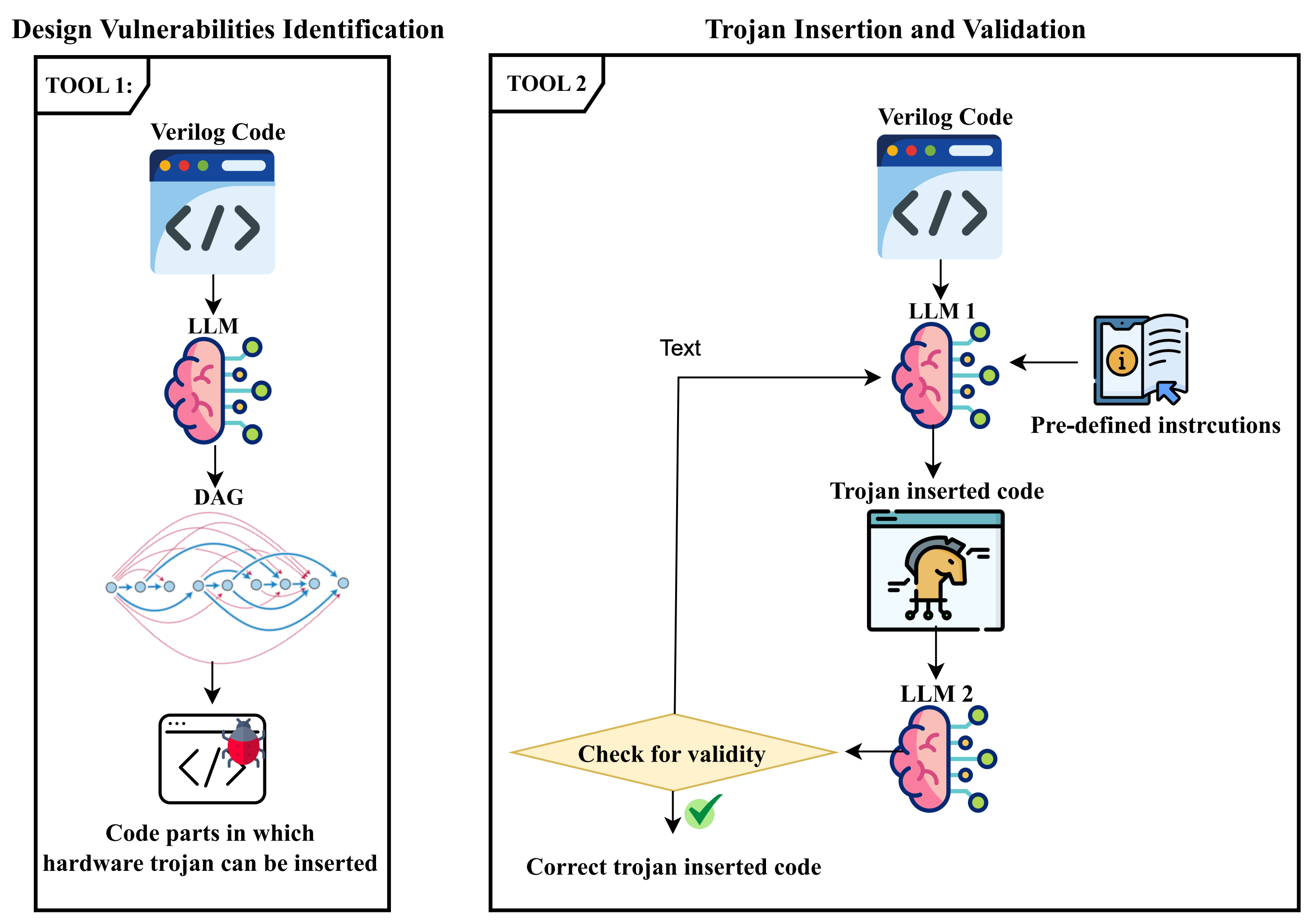
Design Vulnerabilities Identification
Trojan Insertion and Validation
TOOL 1:
Verilog Code
LLM
DAG
Code parts in which hardware trojan can be inserted
TOOL 2
Verilog Code
LLM 1
Text
Pre-defined instrcutions
Trojan inserted code
LLM 2
Check for validity
Correct trojan inserted code

Figure 1. Proposed Methodology

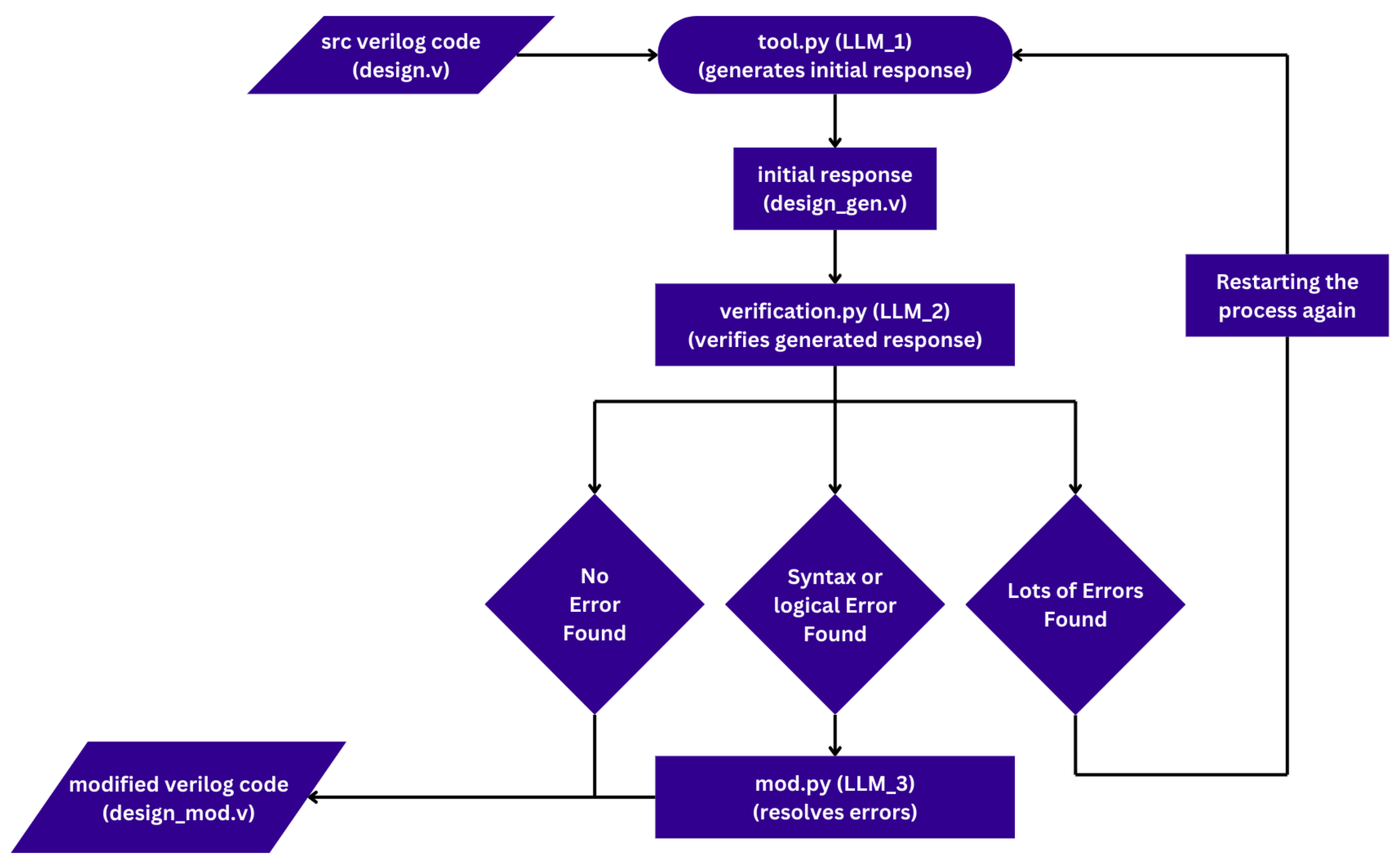
src verilog code (design.v)
tool.py (LLM_1) (generates initial response)
initial response (design_gen.v)
verification.py (LLM_2) (verifies generated response)
Restarting the process again
No Error Found
Syntax or logical Error Found
Lots of Errors Found
modified verilog code (design_mod.v)
mod.py (LLM_3) (resolves errors)

Figure 2. Tool 2 Flowchart: Verilog Code Generation and Validation

Key Features

- The tools are hardware-agnostic and compatible with any Verilog code.
- Users can choose between two leading large language models, Google's Gemini and OpenAI's GPT-4, when initializing the tool.
- We offer both CLI and UI versions of our tool for user convenience.
- An additional validation feature is included to verify the correctness of the code generated by the LLM, which re-queries the LLM if inaccuracies are detected.

Results

Trojan Vulnerabilities in FSM and AES: The FSM and AES examples demonstrate critical vulnerability points for Trojan insertion. FSMs are particularly vulnerable to Trojans due to undefined states or unused transitions, while AES's predictable, key-dependent transformations make it susceptible to subtle interference, such as key schedule manipulation or fault injection in S-boxes.

Tool 1 - Vulnerability Identifier:
Our tool uses a Directed Acyclic Graph (DAG) representation of Verilog code to identify potential vulnerabilities. It then recommends the top five stealthy Trojan insertion strategies tailored to the code structure, helping detect subtle yet impactful Trojans that are challenging to identify manually.

Trojan Insertion in FSM - Bit-Flip and Infinite Loop Trojans:
Two types of Trojans were inserted in an FSM design: a Bit-Flip Trojan, which alters an output signal in the default state, and an Infinite Loop Trojan, which manipulates an unused state, causing the FSM to loop indefinitely. The Infinite Loop Trojan acts as a Denial-of-Service (DoS) attack by making the service unavailable.

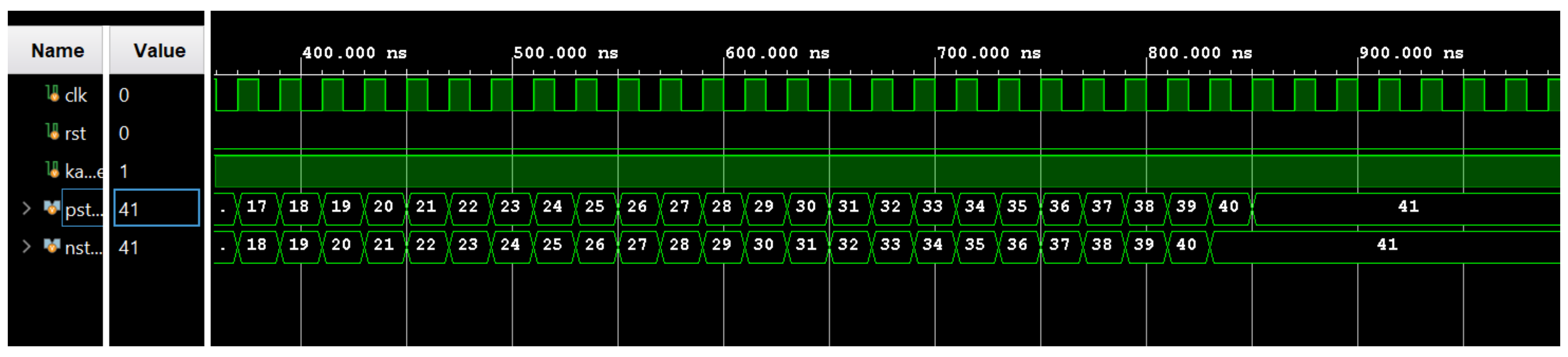

Figure 3. Simulation results of an FSM without Infinite Loop Trojan.

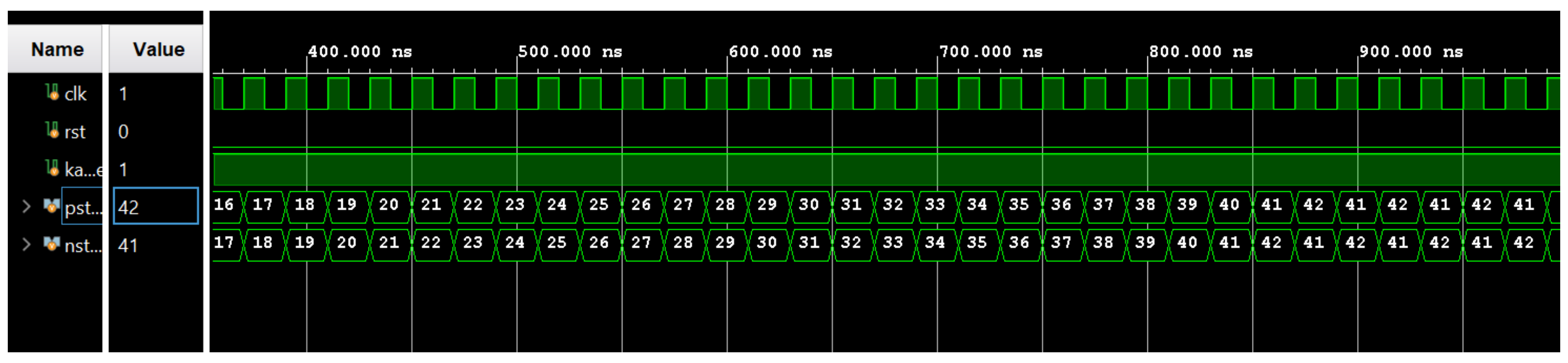

Figure 4. Simulation results of FSM with Infinite Loop Trojan.

Trojan Insertion in AES - Operator Trojan:
In the AES example, a Trojan modifies the MixColumn module and the top AES module. If the input state is even, the MixColumn operation shifts by 2-bits instead of 1-bit, altering the encryption process. The AES top module's encryption success flag is also modified to remain at logic 1, creating the illusion of correct encryption even when it is incorrect.

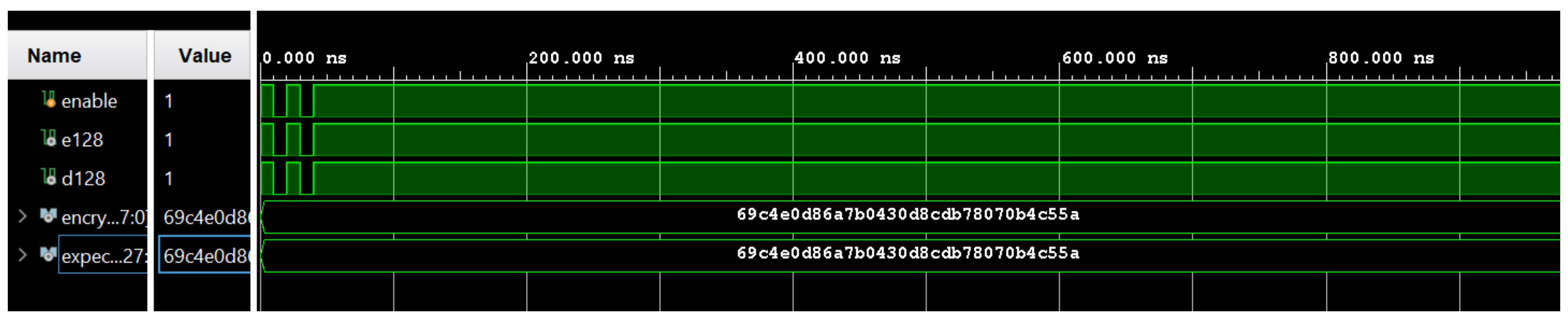

Figure 5. Simulation results of AES operation without Operator Trojan.

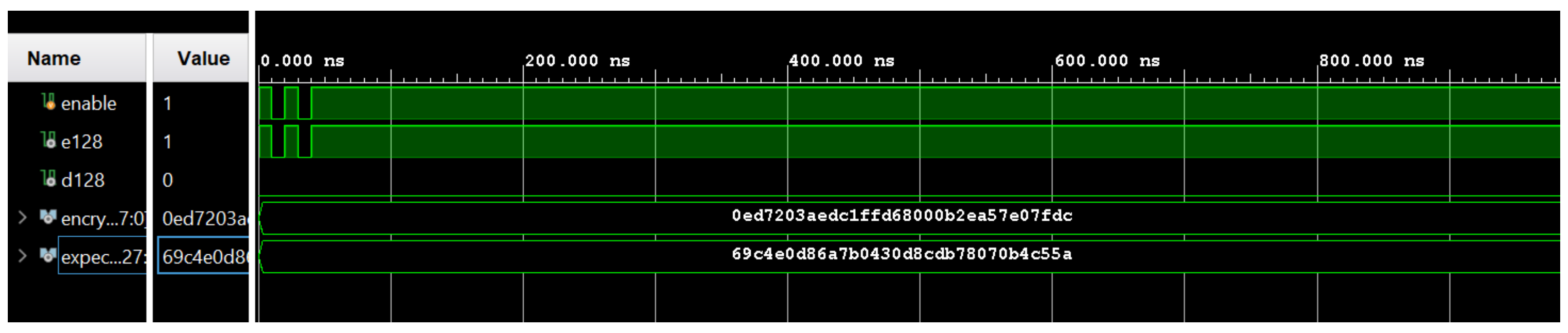

Figure 6. Simulation results of AES operation with Operator Trojan.

Common Vulnerability Scoring System (CVSS) Ratings:
The CVSS score for the Bit-Flip Trojan in FSM is 8, as it affects design integrity by altering outputs undetected. The Infinite Loop Trojan has a CVSS score of 9 due to its significant impact on functionality, effectively causing a denial of service by trapping the FSM in an invalid state.

Conclusions

- We introduce a powerful framework for improving hardware security by automating the identification and insertion of hardware Trojans with the help of advanced large language models.
- Utilizing Google's Gemini and OpenAI's GPT-4, our tools draw on comprehensive knowledge of hardware vulnerabilities to effectively analyze Verilog designs, identify weaknesses, and embed Trojans at optimal insertion points.
- We demonstrate the capabilities of these tools with two applications: Finite State Machines (FSM) and the Advanced Encryption Standard (AES).
- Through inserted trojan in a FSM we illustrate the disruption of control flow in digital circuits and on AES is particularly severe, as it is leading to incorrect encryption, and also bypassing encryption integrity checks.
- Our work showcases the transformative role large language models can play in hardware security, offering a scalable and adaptable approach to assessing complex designs and fortifying defenses against sophisticated threats.