

# Real-time ML-based Defense Against Malicious Payload in Reconfigurable Embedded Systems

Rye Stahle-Smith

University of South Carolina  
Columbia, South Carolina, USA  
ryes@email.sc.edu

Rasha Karakchi (Advisor)

University of South Carolina  
Columbia, South Carolina, USA  
karakchi@cec.sc.edu

## Abstract

The growing use of FPGAs in reconfigurable systems introduces security risks through malicious bitstreams that could cause denial-of-service (DoS), data leakage, or covert attacks. We investigated chip-level hardware malicious payload in embedded systems and proposed a supervised machine learning method to detect malicious bitstreams via static byte-level features. Our approach diverges from existing methods by analyzing bitstreams directly at the binary level, enabling real-time detection without requiring access to source code or netlists. Bitstreams were sourced from state-of-the-art (SOTA) benchmarks and re-engineered to target the Xilinx PYNQ-Z1 FPGA Development Board. Our dataset included 122 samples of benign and malicious configurations. The data were vectorized using byte frequency analysis, compressed using TSVD, and balanced using SMOTE to address class imbalance. The evaluated classifiers demonstrated that Random Forest achieved a macro F1-score of 0.97, underscoring the viability of real-time Trojan detection on resource-constrained systems. The final model was serialized and successfully deployed via PYNQ to enable integrated bitstream analysis.

## Keywords

PYNQ, Machine Learning (ML), AES-128, RS232, Hardware Trojan, Bitstreams, Truncated SVD, SMOTE

## ACM Reference Format:

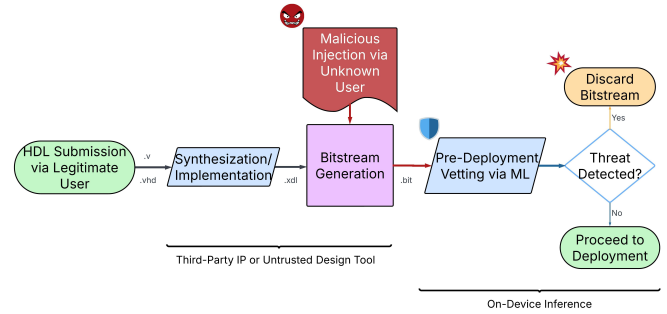
Rye Stahle-Smith and Rasha Karakchi (Advisor). 2025. Real-time ML-based Defense Against Malicious Payload in Reconfigurable Embedded Systems. In *Proceedings of The International Conference for High Performance Computing, Networking, Storage, and Analysis (SC'25)*. St. Louis, MO, USA, 3 pages. <https://doi.org/XXXXXXX.XXXXXXX>

## 1 Introduction

Known for delivering high-performance computation with lower power consumption than GPUs or multicore processors, FPGAs support dynamic updates, making them ideal for adaptive systems, real-time acceleration, and cloud environments. Major providers like Microsoft and IBM have integrated FPGAs into their cloud platforms for applications such as deep learning inference and accelerated networking [5].

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).  
SC'25, St. Louis, MO

© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.  
ACM ISBN 978-1-4503-XXXX-X/25/11  
<https://doi.org/XXXXXXX.XXXXXXX>



**Figure 1: Threat Model for Detecting Trojan-Injected FPGA Bitstreams in Multi-Tenant Systems.**

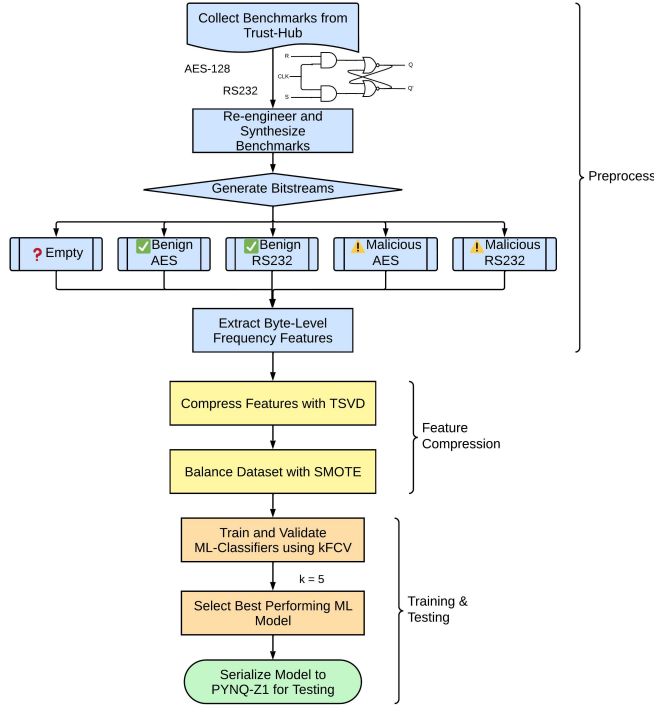
However, this flexibility also opens the door to new attacks. Since FPGAs are configured via binary bitstreams, malicious actors can reprogram devices with compromised data. This risk escalates in multitenant cloud setups, where hardware is shared through time-based reconfiguration, as illustrated in Figure 1. Vendors traditionally mitigate risks by controlling synthesis pipelines or centrally generating bitstreams from submitted IP, but this raises privacy and usability issues, especially when IP confidentiality is critical. In contrast, prior work [2, 12, 14] focuses on detecting hardware Trojans by reverse-engineering bitstreams back into netlists for analysis, which can be time-consuming and reliant on availability of development-stage files.

Our approach diverges by analyzing bitstreams directly at the binary-level, enabling real-time detection without dependency on source or netlist availability. Open hardware initiatives offer transparency and community verification [6], yet hidden vulnerabilities persist, highlighting the need for binary-level detection methods. Prior work has applied machine learning to detect malicious FPGA configurations in shared, multi-tenant settings [5], leveraging structured feature extraction and layered classification to identify complex threats. This project builds on that foundation by targeting embedded environments, where computational overhead and memory are limited. Through streamlined byte-level analysis and efficient data representation, our approach enables real-time detection without reliance on attack-specific heuristics or multi-stage preprocessing. By analyzing bitstream structure, we aim to identify hardware Trojans predeployment.

## 2 Methodology

The general methodology is described in Figures 2 and 3, which illustrate the complete lifecycle of our approach, from offline training and model selection to real-time deployment and inference.

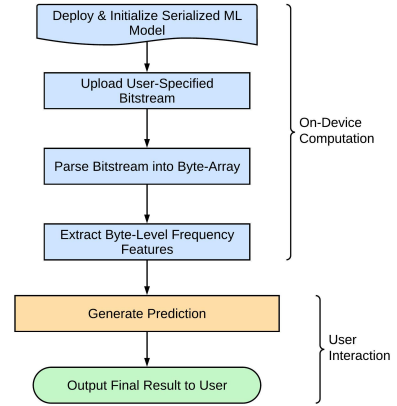
Our approach began with benchmark designs sourced from Trust-Hub [9, 13]. AES-128 and RS232 variants were synthesized into bit files and labeled as benign, malicious, or empty. Byte-level features were extracted from each file, followed by dimensionality reduction using Truncated Singular Value Decomposition (TSVD) to retain structural relevance while improving computational efficiency [11]. To address data set imbalance and model bias [4], the Synthetic



**Figure 2: Training and Deployment Pipeline for Bitstream-Level Trojan Detection.**

Minority Oversampling Technique (SMOTE) was applied to the minority classes in the sample before training [3, 7]. Multiple classifiers were evaluated with the detailed results summarized in Table 1. These classifiers were assessed using k-Fold Cross-Validation (kFCV), a method that divides the dataset into k-subsets, systematically training and testing the model across different folds to robustly estimate performance [10]. This technique often includes mechanisms for parameter tuning and can leverage parallel computation to efficiently select the best-performing model [8].

The best performing model was serialized and deployed to the PYNQ-Z1 [1], where it performed on-device inference using the same preprocessing pipeline to generate predictions. Table 2 summarizes the inference performance and classification results across five deployment trials on the PYNQ-Z1, highlighting the model's consistency and efficiency. This final step validated the feasibility of real-time Trojan detection in a constrained embedded environment without reliance on external computation.



**Figure 3: Runtime Prediction Pipeline on PYNQ-Z1 via Jupyter Notebook UI.**

### 3 Results and Conclusion

The proposed ML-based framework for FPGA bitstream classification achieved strong performance across multiple metrics. As shown in Table 1, the Random Forest classifier reached an accuracy, precision, recall, and F1-score of  $0.98 \pm 0.02$ . On the hold-out test set, it achieved a true positive rate (TPR) of 97.14%, false negative rate (FNR) of 2.86%, false positive rate (FPR) of 0.8%, and an F1-score of 97%.

**Table 1: Performance Metrics (Accuracy, Precision, Recall, F1 Score) for Evaluated Machine Learning Models.**

Model	Accuracy	Precision	Recall	F1 Score
Random Forest	$0.98 \pm 0.02$	$0.99 \pm 0.02$	$0.98 \pm 0.02$	$0.98 \pm 0.02$
Gradient Boosting	$0.90 \pm 0.05$	$0.88 \pm 0.11$	$0.90 \pm 0.06$	$0.88 \pm 0.09$
AdaBoost	$0.89 \pm 0.05$	$0.87 \pm 0.10$	$0.89 \pm 0.05$	$0.87 \pm 0.08$
Logistic Regression	$0.33 \pm 0.07$	$0.14 \pm 0.03$	$0.36 \pm 0.08$	$0.20 \pm 0.05$
Naive Bayes	$0.95 \pm 0.06$	$0.96 \pm 0.05$	$0.96 \pm 0.05$	$0.96 \pm 0.05$
SVM (RBF)	$0.46 \pm 0.10$	$0.30 \pm 0.13$	$0.51 \pm 0.11$	$0.36 \pm 0.12$
KNN	$0.86 \pm 0.04$	$0.87 \pm 0.04$	$0.86 \pm 0.03$	$0.86 \pm 0.04$
Decision Tree	$0.95 \pm 0.00$	$0.96 \pm 0.00$	$0.95 \pm 0.00$	$0.95 \pm 0.00$

The confusion matrix confirmed perfect classification across five classes. The model was successfully deployed on the PYNQ-Z1 platform, with Table 2 showing consistent results over five trials and an average prediction latency of 3.35 seconds. These findings validate the effectiveness of the approach for real-time Trojan detection in resource-constrained environments. Future work will aim to improve robustness against evasive threats and reduce false positives without sacrificing accuracy.

**Table 2: Classification and Timing per Trial**

Trial	Actual Class	Predicted Class	Load (ms)	Extract (ms)	Predict (ms)
1	Mal. RS232	Mal. RS232	239.48	3157.72	15.03
2	Mal. RS232	Mal. RS232	190.72	3268.73	16.94
3	Ben. AES	Ben. AES	23.65	3167.01	16.81
4	Ben. AES	Ben. AES	21.85	3156.55	16.96
5	Mal. AES	Mal. AES	189.02	3228.77	15.85

## 4 Acknowledgments

This work was supported by the McNair Junior Fellowship and OUR at the University of South Carolina. This project used benchmark designs from Trust-Hub, a resource sponsored by the National Science Foundation (NSF). OpenAI's ChatGPT was employed to help refine language and grammar. All technical content and analysis were developed independently by the authors. This research also made use of the PYNQ-Z1 FPGA platform, provided by AMD and Xilinx, whose tools and hardware enabled the synthesis and deployment stages of this study.

## References

- [1] AMD (2024). PYNQ™: Python Productivity for Zynq. <https://www.pynq.io>. Accessed: Aug. 4, 2025.
- [2] Benz, F., Seffrin, A., and Huss, S. A. (2012). Bil: A tool-chain for bitstream reverse-engineering. In *Proceedings of the IEEE International Conference on Field Programmable Logic and Applications (FPL)*, pages 735–738. IEEE.
- [3] Chawla, N., Bowyer, K., Hall, L., and Kegelmeyer, W. (2002). Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357.
- [4] Elnaggar, R. and Chakrabarty, K. (2018). Machine learning for hardware security: Opportunities and risks. *Journal of Electronic Testing*, 34(2):183–201.
- [5] Elnaggar, R., Chaudhuri, J., Karri, R., and Chakrabarty, K. (2023). Learning malicious circuits in fpga bitstreams. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 42(3):726–739.
- [6] Hayashi, V. T. and Ruggiero, W. V. (2025). Hardware trojan detection in open-source hardware designs using machine learning. *IEEE Access*.
- [7] Imbalanced-learn Developers (2024). Smote. <https://bit.ly/3IXc0l7>. Accessed: Aug. 4, 2025.
- [8] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830.
- [9] Salmani, H., Tehranipoor, M., and Karri, R. (2013). On design vulnerability analysis and trust benchmark development. In *Proceedings of the IEEE International Conference on Computer Design (ICCD)*. IEEE.
- [10] Scikit-learn Developers (2025a). Cross validation. <https://bit.ly/3gct8QG>. Accessed: Aug. 4, 2025.
- [11] Scikit-learn Developers (2025b). Truncated svd. <https://bit.ly/4mmi4BT>. Accessed: Aug. 4, 2025.
- [12] Seo, Y., Yoon, J., Jang, J., Cho, M., Kim, H.-K., and Kwon, T. (2018). Poster: Towards reverse engineering fpga bitstreams for hardware trojan detection. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, pages 18–21. Internet Society.
- [13] Shakya, B., He, T., Salmani, H., Forte, D., Bhunia, S., and Tehranipoor, M. (2017). Benchmarking of hardware trojans and maliciously affected circuits. *Journal of Hardware and Systems Security*.
- [14] Yoon, J., Seo, Y., Jang, J., Cho, M., Kim, J., Kim, H., and Kwon, T. (2018). A bitstream reverse engineering tool for fpga hardware trojan detection. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS '18*, pages 2318–2320, New York, NY, USA. Association for Computing Machinery.