An FPGA-Accelerated Decision Tree Model for Ransomware Detection in Pervasive Environments

Camilly de Melo Cruvinel[1], Radharani Santos Rocha[2], Camilla Charão Borchhardt Quincozes[3], Diego Nunes Molinos[4]

[1]Faculdade de Computação (FACOM) – Universidade Federal de Uberlândia (UFU)
Uberlândia, MG - Brasil

{camilly.cruvinel,radharanisr,camillaquincozes,diego.molinos}@ufu.br

Abstract. *Ransomware poses a threat to pervasive computing environments, where resource-constrained devices require low-latency detection mechanisms. This work presents an FPGA-based hardware classifier that translates a trained decision tree into a fixed-point combinational architecture, enabling deterministic single-cycle inference. The proposed architecture achieves up to 99% accuracy with a constant latency of 20 ns per sample. Compared to a software-based ML implementation, the FPGA solution delivers speedups of up to* 2.46×10^3 *while utilizing less than 1% of the available logic elements, demonstrating the feasibility of ransomware detection in resource-constrained environments.*

1. Introduction

Pervasive computing integrates computational devices into the physical environment, enabling sensors, embedded systems, and mobile devices to operate seamlessly in connected environments [Haque and Ahamed 2006]. However, this paradigm also expands the attack surface, particularly in resource-constrained devices.

Among emerging threats, ransomware stands out for its operational impact on data and service availability, affecting critical infrastructure [Humayun et al. 2021]. The effectiveness of this type of attack depends on the encryption speed, making early-stage detection essential to prevent irreversible damage [Malik et al. 2024].

In response to the evolution of ransomware attacks, Machine Learning (ML)-based approaches have been widely explored for automated threat detection. However, most existing solutions rely on high-level frameworks and run on conventional operating systems, which introduces computational overhead, increased inference latency, and higher energy consumption. These characteristics limit their applicability in pervasive environments, where computational, energy, and storage resources are constrained, and real-time detection is critical for promptly mitigating attacks.

In this context, *Field-Programmable Gate Arrays* (FPGAs) emerge as a promising alternative, enabling dedicated hardware execution that reduces inference latency and improves energy efficiency compared to software-based solutions [Ngo et al. 2021]. Their inherent parallelism and reconfigurability allow detection mechanisms to run directly on the device, reducing dependence on complex operating systems and frameworks.

Despite recent advances in ransomware detection, pervasive environments introduce new challenges, including the need for lightweight, low-latency detection mechanisms on resource-constrained devices. Therefore, this work presents an FPGA-based

architecture that materializes a trained decision tree as a hardware-oriented classifier for ransomware detection in pervasive systems. The proposed implementation achieves deterministic single-cycle inference with very low hardware utilization, enabling low-latency detection on resource-constrained devices.

This work is guided by three research questions: **RQ1:** Can FPGA-based architectures enable ransomware detection with low latency and minimal hardware resources? **RQ2:** How much latency reduction can hardware inference provide compared to software implementation? **RQ3:** What are the trade-offs between classification accuracy and computational performance in reconfigurable hardware implementations?

2. Background and Related Work

This section reviews security challenges in pervasive computing, ML approaches to ransomware detection, and FPGA-based acceleration for low-latency on-device security.

2.1. Ransomware in Pervasive Computing Environments

Pervasive computing integrates computational capabilities into everyday environments, enabling sensors, embedded systems, and mobile devices to operate continuously [Satyanarayanan 2002]. These environments typically include persistent file systems and exposed network interfaces, increasing their susceptibility to malware, particularly ransomware attacks [Park et al. 2022, Azmoodeh et al. 2018]. In this context, ransomware has evolved from attacks targeting personal computers to a broader threat affecting IoT, mobile, and cyber-physical environments [Oz et al. 2022].

This evolution is evidenced by platform-specific variants such as Android and Linux ransomware [Lipovskỳ et al. 2016, Mohurle and Patil 2017]. Recent campaigns, including *Ransomware-as-a-Service*, demonstrate a shift toward platform-agnostic and service-oriented attack strategies [Kusuma et al. 2021, Alwashali et al. 2021]. As pervasive environments integrate multiple components, ransomware has become a cross-platform threat that affects digital assets and operational services.

2.2. Ransomware Detection using Machine Learning

Traditional ransomware detection relies on signature-based techniques that identify malware through known patterns or predefined code signatures [Alzahrani et al. 2025]. Although effective against known threats, these approaches struggle to detect new variants due to obfuscation and polymorphism, limiting their ability to identify zero-day attacks [Alraizza and Algarni 2023].

Machine learning has emerged as an alternative approach that detects behavioral patterns rather than static signatures. However, many ML-based solutions depend on complex frameworks and significant computational resources, limiting their deployment in resource-constrained environments [Alraizza and Algarni 2023].

2.3. FPGA-based Machine Learning Acceleration

In ransomware detection scenarios, large amounts of behavioral data must be analyzed under strict latency constraints. Such workloads often involve irregular memory access patterns and data-dependent execution, reducing efficiency in general-purpose architectures [Gajjar et al. 2024]. In this context, using FPGA-based acceleration appears to be

a promising alternative. FPGAs enable hardware-level reconfigurability and exploit fine-grained parallelism, allowing architectures to be tailored for machine learning inference workloads [Makrani et al. 2022, Ngo et al. 2021]. These characteristics are particularly suitable for pervasive environments, enabling efficient execution while reducing computational overhead and improving energy efficiency.

2.4. Related Work

Table 1 summarizes representative studies exploring FPGA-based acceleration for ML models. Several FPGA-based approaches focus on achieving high throughput for tree-based machine learning inference in enterprise or data center environments, typically employing multi-stage pipelines, multiple compute units, and complex memory hierarchies. Although effective for large workloads, such architectures often require significant hardware resources.

Table 1. Comparison of FPGA-based acceleration approaches

Work	ML Model	Platform	Hardware Design	Main Contribution
[Gajjar et al. 2024]	XGBoost	FPGA	Multi-stage pipeline	High-throughput FPGA accelerator for tree-based ML inference
[Alcolea and Resano 2021]	GBDT	FPGA	Multi-threaded class	Energy-efficient FPGA accelerator for gradient boosting trees
[Kulaga and Gorgon 2014]	Ensembles	FPGA (Zynq)	Parallel pipeline	Parallel tree evaluation for improved inference throughput
[Anand et al. 2023]	Random Forest	CPU	HPC monitoring	Early ransomware detection using hardware performance counters
This work	Decision Tree	FPGA	Combinational classifier	Single-cycle FPGA classifier for ransomware detection

In contrast, this work targets ransomware detection in pervasive environments, prioritizing lightweight hardware design and deterministic low-latency inference suitable for resource-constrained devices.

3. Methodology

This research adopts a design-oriented experimental methodology in which a solution is conceived, implemented, and evaluated to assess improvements over existing approaches. Figure 1 illustrates the methodological workflow adopted in this work.

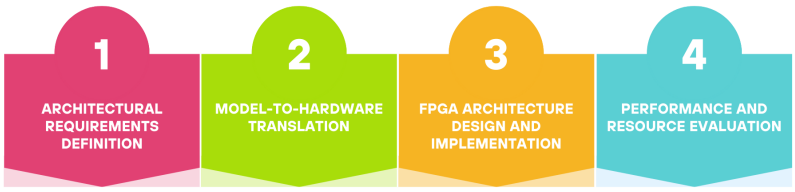

Figure 1. Overview of the methodological workflow

3.1. Architectural Requirements Definition

Initially, a literature review was conducted to examine existing FPGA-based acceleration approaches. The analysis identified architectural patterns and implementation strategies for accelerating machine learning inference on hardware platforms.

3.2. Ransomware Dataset and ML Classifier Setup

The CIC-MalMem-2022 dataset [Canadian Institute for Cybersecurity 2022] was used to train the classifier later translated into FPGA hardware to classify ransomware samples. This dataset contains behavioral memory features extracted from controlled executions of benign applications and obfuscated malware samples, comprising 58,596 instances, equally divided between benign and malicious samples, with 29,298 records per category. In this study, all ransomware samples available in the dataset were utilized without excluding any subtypes. The ransomware category includes samples from the *Conti, MAZE, Pysa, Ako*, and *Shade*.

The classifier is based on supervised machine learning, using a decision tree classifier trained to distinguish between malicious and benign behaviors. A 70/30 train-test split was employed, along with a 5-fold, to ensure consistency and minimize overfitting. The classifier was trained using the default parameters provided by the adopted machine learning library.

3.3. FPGA Architecture Design and Implementation

The architecture follows a hardware-oriented model in which input features are evaluated through a hierarchical sequence of comparisons that mirrors the structure of the trained decision tree. Features are represented using 32-bit fixed-point signed arithmetic, enabling efficient numerical comparisons without the overhead of floating-point operations. Threshold values extracted from the trained model were embedded as constants in the VHDL description, allowing the classifier to operate primarily as combinational comparison logic followed by a clocked output register.

3.4. Performance and Resource Evaluation

Experiments were conducted using subsets of the CIC-MalMem-2022 dataset, which included both benign applications and ransomware samples from the Conti, MAZE, Pysa, Ako, and Shade families. The subsets contained 1, 50, 100, 150, 500, and 1000, including both malicious and benign random samples. To ensure fair and reproducible comparisons, the same input instances were used for both the ML classifier and FPGA validation.

For each configuration, we evaluated performance metrics such as inference latency, throughput, CPU utilization, memory consumption, and hardware resource usage. The ML classifier was validated using a 70/30 train-test split and 5-fold cross-validation to assess its consistency. Meanwhile, the FPGA evaluation focused on deterministic hardware inference performance.

4. Model-to-Hardware Design

The proposed design translates the ML classifier, trained decision tree, into an FPGA inference structure. The Decision nodes are implemented as fixed-point comparators forming combinational logic that reproduces the model's hierarchical evaluation. Using 32-bit signed fixed-point arithmetic enables deterministic, low-latency inference with minimal hardware resources.

4.1. Decision Tree Model - ML classifier

A supervised decision tree model was trained to distinguish benign from ransomware behaviors. The training results are summarized in Table 2, which shows that 5 fold cross-validation achieves a mean accuracy of 0.9996. The final test evaluation reached weighted precision, F1-score, recall, and accuracy of 0.99949, while Figure 2 presents the learning curves and confusion matrix.

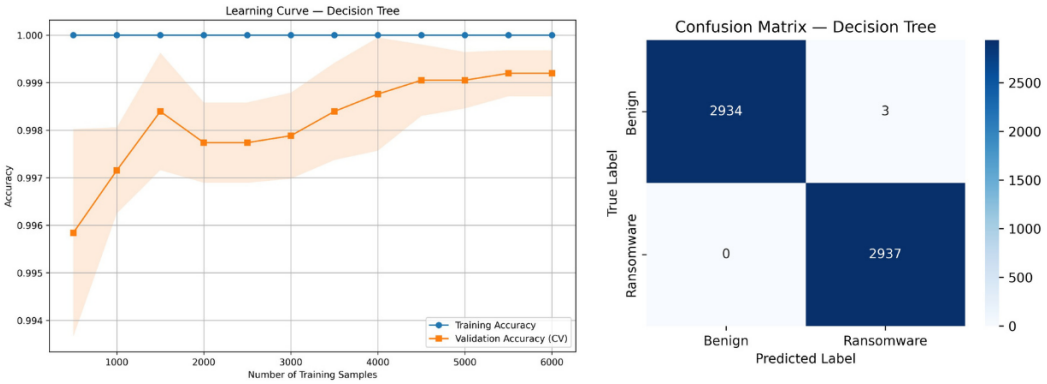

Figure 2. Learning Curve and Confusion Matrix - Decision Tree

Table 2. Decision Tree Model Performance

Cross-Validation Accuracy						Test Metrics			
Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean	Precision	F1-score	Accuracy	Recall
0.9996	0.9993	1.0000	0.9996	0.9996	0.9996	0.9995	0.9995	0.9995	0.9994

Although models such as Random Forests, Gradient Boosting, MLPs, and XGBoost are adopted in the literature for ransomware detection, this work employs a decision tree classifier. This choice was intentional, as decision trees have low structural complexity, are highly interpretable, and can be directly mapped to hardware. These features are crucial for deployment on FPGAs, where consistent inference latency and efficient hardware utilization are important design constraints.

After training, the ML classifier was analyzed to extract decision rules and thresholds for the hardware-oriented classifier. Figure 3 shows the resulting model decision used for the FPGA implementation. Although metrics such as ROC curves and AUC could also be reported, the purpose of this work is not to discuss specific characteristics of the ML classifier itself. Instead, the focus is on evaluating the feasibility of translating a trained classifier into an FPGA-based implementation.

4.2. FPGA-Based Hardware-Oriented Classifier Architecture

The Figure 4 illustrates the hardware architecture derived from the trained decision tree model and implemented on the FPGA. The proposed ML classifier is implemented as a combinational datapath that performs decision-tree inference via parallel fixed-point threshold comparisons. Unlike software implementations, the FPGA design exploits spatial parallelism to evaluate multiple feature comparisons simultaneously.

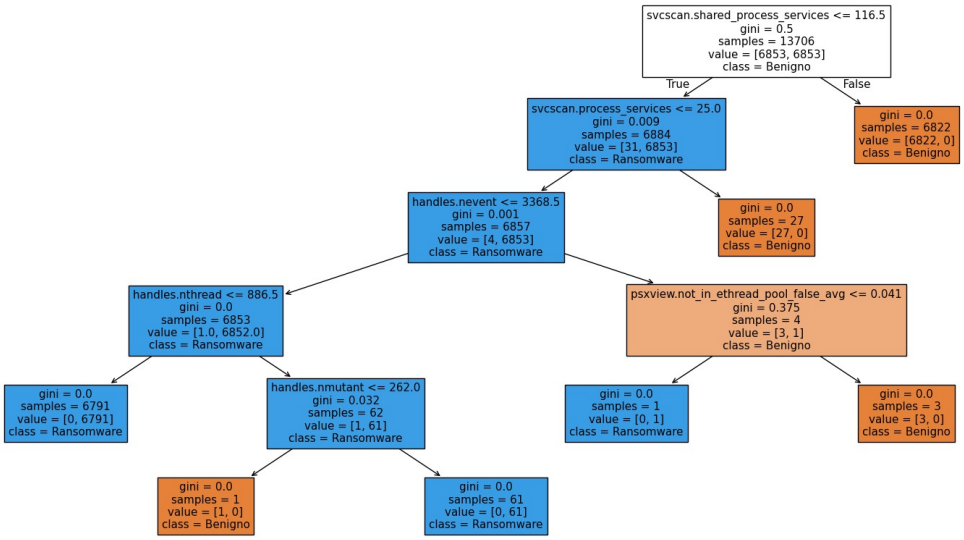

Figure 3. the resulting decision tree model

The model's thresholds are embedded as fixed-point constants, eliminating floating-point operations and iterative control logic. Each input feature is compared against its threshold using dedicated comparators, and the resulting signals are processed by a decision module to generate the final classification. This design forms a compact hardware structure that enables deterministic, low-latency inference.

Although inference is implemented using combinational logic, it is still constrained by synchronous timing requirements. The hardware-oriented classifier output depends on signal propagation through the combinational datapath composed of the subtractor and comparators (*C1–C6*). For correct operation, the clock period must satisfy $T_{clk} \geq T_{comb} + T_{setup} + T_{skew} + T_{jitter}$, where T_{comb} represents the critical-path delay, T_{setup} the register setup time, and T_{skew} and T_{jitter} the clock distribution uncertainties. Thus, the clock frequency determines the maximum throughput.

The proposed hardware-oriented classifier has constant-time execution complexity $O(1)$, where inference latency is independent of dataset size or input distribution. The architecture evaluates decision conditions in parallel using combinational logic, enabling single-cycle classification with deterministic latency defined by the hardware critical path.

4.3. Hardware Mapping

To demonstrate the model-to-hardware translation process, each decision node of the trained decision tree is mapped to a fixed-point comparison structure in VHDL. For example, the decision rule "if feature_3 $\leq$ 0.42 then benign else ransomware" is transformed into a digital comparison operation using 32-bit signed fixed-point arithmetic. The threshold values from the trained model are embedded as constants in the hardware description, enabling direct synthesis of LUT-based comparators on the FPGA fabric.

These comparators' outputs are combined according to the tree's structure to generate the final classification signal. The hardware-oriented classifier leverages parallelism, allowing multiple comparison paths to be evaluated simultaneously, which results in deterministic single-cycle inference with minimal overhead.

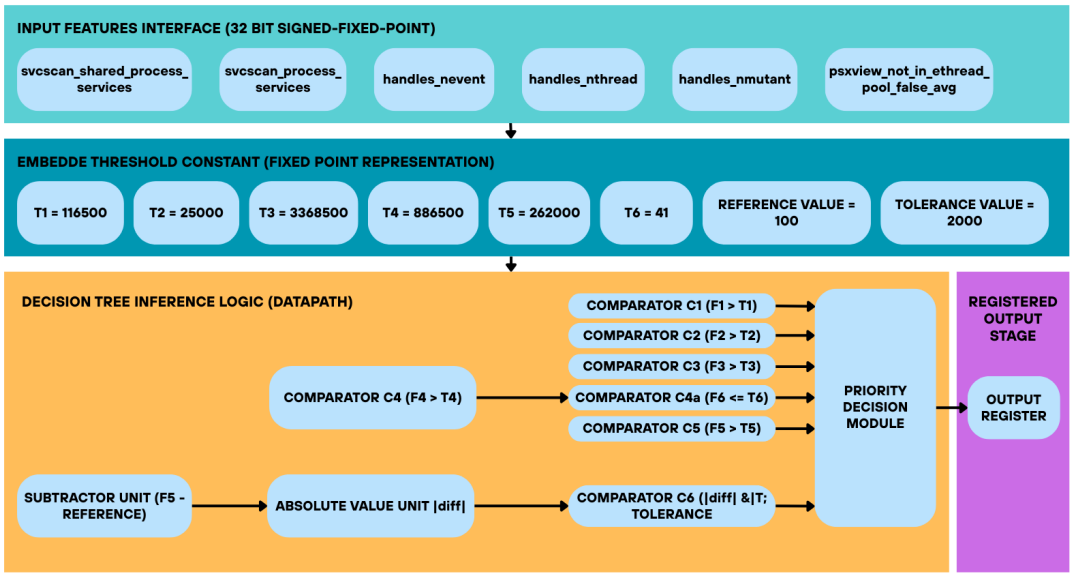

Figure 4. Overview of an FPGA-Based Hardware-Oriented Classifier Architecture

Table 3 shows an illustrative example of how ML classifier rules are translated into hardware-oriented classifier structures using fixed-point comparison logic and combinational datapaths.

Table 3. Example of Decision Tree Node Translation to FPGA Logic

Decision Tree Rule	Hardware Representation	FPGA Mapping
if $feature_3 \leq 0.42$	Comparator operation	LUT-based fixed-point comparator
threshold = 0.42	Fixed-point constant	Embedded constant in VHDL
benign / ransomware output	Binary classification signal	Registered output logic
hierarchical decision path	Conditional evaluation chain	Combinational datapath
multiple node evaluations	Parallel comparisons	Spatial parallelism in FPGA

5. Experiments and Results Analysis

This experiment compares the computational overhead of the inference pipeline in a software runtime and a dedicated FPGA implementation, considering latency, CPU utilization, power consumption, and memory footprint. The first n instances of the test dataset were used across all implementations to ensure reproducibility.

Software experiments were conducted on a Fedora Linux 41 workstation with an Intel Core i5-7200U (2.50 GHz) using Python. The hardware implementation targeted an Intel Cyclone IV FPGA (EP4CE115F29C7) on an Altera DE2-115 board at 50 MHz, IDE Quartus Prime Lite Edition 22.1 with functional verification performed in ModelSim.

5.1. ML Classifier Baseline Performance

A warm-up was executed before measurements to stabilize the execution environment, and these runs were excluded from the reported results. Table 4 summarizes the performance metrics of the ML classifier inference pipeline. The results show that total inference latency remains relatively stable across batch sizes, indicating a fixed runtime

overhead in the execution environment. The standard deviation is included to capture the runtime variability inherent to ML classifier execution in a general-purpose operating system.

Table 4. Software Inference Performance Metrics

Batch	Avg. Lat. (s)	Std. Dev. (s)	Time/Inst. (s)	Throughput (s)	CPU (%)	Mem (MB)
1	0.003981	0.000905	0.003981	251	88.58	336.89
50	0.003750	0.000516	0.000075	13333	103.21	336.89
100	0.003372	0.000824	0.000034	29650	103.45	336.89
150	0.002638	0.000870	0.000018	56800	116.75	336.89
500	0.002255	0.000657	0.0000045	221700	83.29	336.89
1000	0.003489	0.000872	0.0000035	286000	78.25	337.28

CPU utilization remained high during ML classifier inference. Table 4 shows average usage across batch sizes, where values above 100% indicate multi-core utilization, which may be problematic in resource-constrained pervasive environments. The memory footprint remained stable across batch sizes at approximately 337 MB (Table 4), indicating that most consumption comes from the runtime environment and model structures, which may be unsuitable for resource-constrained devices.

5.2. Hardware-Oriented Classifier Implementation Results

The Table 5 summarizes the hardware-oriented classifier resource utilization of the proposed architecture on the target FPGA device.

Table 5. FPGA Resource Utilization Summary

Resource	Used	Available	Utilization
Logic Elements	146	114,480	$<1\%$
Registers	1	–	–
Pins	195	529	37%
Virtual Pins	0	–	0%
Memory Bits	0	3,981,312	0%
Embedded Multipliers (9-bit)	0	532	0%
PLLs	0	4	0%

Power consumption was estimated using the Quartus Prime Power Analyzer after place-and-route, resulting in a total thermal dissipation of approximately 176.6 mW (98.61 mW core static power and 77.99 mW I/O power). As switching activity was not included, the estimate mainly reflects baseline device power, indicating minimal additional overhead from the proposed classifier. Considering the single-cycle latency of 20 ns at 50 MHz and the estimated power dissipation of 176.6 mW, the energy consumption per inference is approximately 3.5 nJ.

5.3. Performance Comparison

The main architectural differences between the ML classifier and the hardware-oriented classifier are summarized in Table 6. While the ML classifier runs on a general-purpose CPU and depends on the operating system and runtime environment, the hardware-oriented classifier uses dedicated FPGA combinational logic.

Table 6. Software vs. FPGA Inference Resource Perspective

Metric	Software Runtime	FPGA
Execution model	General-purpose CPU	Dedicated hardware logic
Avg. inference latency	33–49 μs	20 ns
CPU utilization	78.25–116.75%	Not applicable (hardware execution)
Resident memory footprint	336.89–337.28 MB	No dedicated memory blocks
Logic elements	N/A	146
DSP / multipliers	N/A	0
Clock frequency	CPU-dependent	50 MHz
Determinism	OS/runtime-dependent	Deterministic hardware latency

The ML classifier achieves inference times of 33–49 μs per sample, while the hardware-oriented classifier performs inference in a single clock cycle with approximately 20 ns latency, using only 146 logic elements and no DSP blocks. Table 6 shows scalability across workloads from 1 to 1000 samples. While the ML classifier benefits from overhead amortization as the workload increases, the hardware-oriented classifier maintains constant-time inference per sample, resulting in linear scaling of total execution.

Table 7. Inference Latency and Throughput Comparison for Different Workloads

Instances	Implementation	Latency / sample	Total Time	Speedup
1	Software ML	0.003981 s	0.003981 s	1×
	FPGA (50 MHz)	20 ns	20 ns	$\approx 2.0 \times 10^5$
50	Software ML	49.2 μs	2.46 ms	1×
	FPGA (50 MHz)	20 ns	1 μs	$\approx 2.46 \times 10^3$
100	Software ML	38.97 μs	3.897 ms	1×
	FPGA (50 MHz)	20 ns	2 μs	$\approx 1.95 \times 10^3$
150	Software ML	33.44 μs	5.016 ms	1×
	FPGA (50 MHz)	20 ns	3 μs	$\approx 1.67 \times 10^3$
500	Software ML	4.5 μs	2.255 ms	1×
	FPGA (50 MHz)	20 ns	10 μs	$\approx 2.25 \times 10^2$
1000	Software ML	3.5 μs	3.489 ms	1×
	FPGA (50 MHz)	20 ns	20 μs	$\approx 1.74 \times 10^2$

Despite linear scaling, the hardware-oriented classifier consistently outperforms the ML classifier, achieving speedups of up to 2×10^5 for small workloads and remaining on the order of 10^2–10^3 for larger batches. Table 8 summarizes the average classification performance across workloads (1–1000 samples). The hardware-oriented classifier achieves 98.33% accuracy, with 99.00% precision and 97.55% recall. Although slightly lower than the ML classifier (approx. 99.95% accuracy), it achieves comparable predictive performance while incurring significantly lower inference latency.

Finally, this evaluation focuses on architectural feasibility and hardware efficiency rather than full system deployment. Therefore, synthesis results and static timing analysis are sufficient to demonstrate the viability of the proposed hardware-oriented classifier for latency-critical embedded applications.

Table 8. Overall Comparison Between Software and FPGA Implementations

Implementation	Accuracy	Precision	Recall	Latency (Max)
Software ML (Average)	99.95%	99.95%	99.94%	49.2 μs
FPGA (Average)	98.33%	99.00%	97.55%	20 ns

5.4. Discussion and Practical Implications

The experimental results highlight an important trade-off between predictive performance and computational efficiency. Although the hardware-oriented classifier presents a slight reduction in classification accuracy compared to the ML classifier, mainly due to the use of fixed-point arithmetic and hardware-oriented logical representations instead of floating-point computation, it enables substantial gains in computational efficiency, including deterministic nanosecond-scale inference, minimal hardware resource utilization, reduced memory requirements, and low energy consumption. These characteristics make the proposed architecture particularly suitable for pervasive and resource-constrained environments.

In real-world applications, challenges extend beyond inference latency. The hardware-oriented classifier must integrate into a continuous monitoring pipeline, requiring efficient feature extraction, memory transfer, and synchronization with operating system events or file monitors. Real-time environments can create I/O and streaming throughput bottlenecks while also addressing dynamic workload variations. Additionally, updating models can necessitate partial or full hardware reconfiguration. Nonetheless, the proposed architecture shows that low-latency hardware inference is feasible in resource-constrained environments.

6. Conclusion

This work presented an FPGA architecture for ransomware detection in pervasive environments by translating a trained decision tree into a hardware-oriented classifier. The proposed design enables deterministic single-cycle inference using lightweight combinational logic. Regarding RQ1, the results indicate that a hardware-oriented classifier can achieve efficient, low-latency ransomware detection with minimal hardware resources, using only 146 logic elements and no DSP blocks. Regarding RQ2, hardware inference significantly reduces latency compared to the software baseline, achieving approximately 20 ns per sample versus 33–49 μs in software, corresponding to speedups of up to five orders of magnitude. Regarding RQ3, the hardware-oriented classifier preserves comparable predictive performance, achieving 98.33% accuracy while offering substantial computational efficiency gains. The use of fixed-point arithmetic slightly alters the numerical representation compared to floating-point implementations (ML-based), introducing a minor variation in classification accuracy while significantly reducing hardware complexity. These results demonstrate the feasibility of deploying lightweight classifiers on FPGAs for low-latency security in pervasive environments. Future work includes evaluating ensemble-based approaches, such as Random Forests and Gradient Boosting, on FPGA architectures and investigating trade-offs among predictive accuracy, hardware complexity, resource utilization, and inference latency.

References

Alcolea, A. and Resano, J. (2021). Fpga accelerator for gradient boosting decision trees. *Electronics*, 10(3):314.

Alraizza, A. and Algarni, A. (2023). Ransomware detection using machine learning: A survey. *Big Data and Cognitive Computing*, 7(3):143.

Alwashali, A. A. M. A., Abd Rahman, N. A., and Ismail, N. (2021). A survey of ransomware as a service (raas) and methods to mitigate the attack. In *2021 14th International Conference on Developments in eSystems Engineering (DeSE)*, pages 92–96. IEEE.

Alzahrani, S., Xiao, Y., Asiri, S., Zheng, J., and Li, T. (2025). A survey of ransomware detection methods. *IEEE Access*.

Anand, P. M., Charan, P. S., and Shukla, S. K. (2023). Hiper-early detection of a ransomware attack using hardware performance counters. *Digital Threats: Research and Practice*, 4(3):1–24.

Azmoodeh, A., Dehghantanha, A., Conti, M., and Choo, K.-K. R. (2018). Detecting crypto-ransomware in iot networks based on energy consumption footprint. *Journal of Ambient Intelligence and Humanized Computing*, 9(4):1141–1152.

Canadian Institute for Cybersecurity (2022). CIC-MalMem-2022 Dataset. `https://www.unb.ca/cic/datasets/malmem-2022.html`. Accessed: 10-10-2025.

Gajjar, A., Kashyap, P., Aysu, A., Franzon, P., Choi, Y., Cheng, C., Pedretti, G., and Ignowski, J. (2024). Rd-faxid: Ransomware detection with fpga-accelerated xgboost. *ACM Transactions on Reconfigurable Technology and Systems*, 17(4):1–33.

Haque, M. and Ahamed, S. I. (2006). Security in pervasive computing: Current status and open issues. *International Journal of Network Security*.

Humayun, M., Jhanjhi, N. Z., Alsayat, A., and Ponnusamy, V. (2021). Internet of things and ransomware: Evolution, mitigation and prevention. *Egyptian Informatics Journal*, 22(1):105–117.

Kulaga, R. and Gorgon, M. (2014). Fpga implementation of decision trees and tree ensembles for character recognition in vivado hls. *Image Processing & Communications*, 19(2-3):71.

Kusuma, R. S., Umar, R., and Riadi, I. (2021). Network forensics against ryuk ransomware using trigger, acquire, analysis, report, and action (taara) method. *Kinetik: Game Technology, Information System, Computer Network, Computing, Electronics, and Control*.

Lipovskỳ, R., Štefanko, L., and Braniša, G. (2016). The rise of android ransomware.

Makrani, H. M., He, Z., Rafatirad, S., and Sayadi, H. (2022). Accelerated machine learning for on-device hardware-assisted cybersecurity in edge platforms. In *2022 23rd International Symposium on Quality Electronic Design (ISQED)*, pages 77–83. IEEE.

Malik, V., Khanna, A., Sharma, N., et al. (2024). Trends in ransomware attacks: analysis and future predictions. *International Journal of Global Innovations and Solutions (IJGIS)*.

Mohurle, S. and Patil, M. (2017). A brief study of wannacry threat: Ransomware attack 2017. *International journal of advanced research in computer science*, 8(5).

Ngo, D.-M., Temko, A., Murphy, C. C., and Popovici, E. (2021). Fpga hardware acceleration framework for anomaly-based intrusion detection system in iot. In *2021 31st International conference on field-programmable logic and applications (FPL)*, pages 69–75. IEEE.

Oz, H., Aris, A., Levi, A., and Uluagac, A. S. (2022). A survey on ransomware: Evolution, taxonomy, and defense solutions. *ACM Computing Surveys (CSUR)*, 54(11s):1–37.

Park, J. H., Singh, S. K., Salim, M. M., Azzaoui, A. E., and Park, J. H. (2022). Ransomware-based cyber attacks: A comprehensive survey. *Journal of Internet Technology*, 23(7):1557–1564.

Satyanarayanan, M. (2002). Pervasive computing: Vision and challenges. *IEEE Personal communications*, 8(4):10–17.