ISSN: 2395-1680 (ONLINE)
DOI: 10.21917/ijme.2024.0288

ENHANCING AI MODEL SECURITY USING A HARDWARE-BASED APPROACH FOR PROTECTING FPGA IMPLEMENTATION

Syed Arfath Ahmed[1], R.K. Agrawal[2], Neelam Labhade Kumar[3], V.S. Narayana Tinnaluri[4] and Geogen George[5]

[1]*Department of Computer Science and Engineering, Maulana Azad National Urdu University, India*
[2]*Department of Electronics and Telecommunication Engineering, SNJB's Late Sau Kantabai Bhavarlalji Jain College of Engineering, India*
[3]*Department of Computer Science and Engineering, Shree Ramchandra College of Engineering, India*
[4]*Department of Computer Science, Koneru Lakshmaiah Education Foundation, India*
[5]*Department of College of Computing and Information Sciences, University of Technology and Applied Sciences, Sultanate of Oman*

Abstract

In the ever-evolving landscape of artificial intelligence (AI), the vulnerability of AI models to adversarial attacks has become a critical concern. The problem at hand lies in the lack of dedicated security mechanisms tailored for FPGA-based AI implementations, leaving them exposed to threats such as tampering, reverse engineering, and unauthorized access. This research addresses the pressing need for robust security measures by proposing a hardware-based approach to protect FPGA (Field-Programmable Gate Array) implementations of AI models. FPGAs offer a flexible and efficient platform for deploying AI models but are susceptible to attacks that compromise the integrity of the implemented algorithms. This involves the integration of specialized security modules within the FPGA architecture. These modules are designed to detect and thwart various forms of attacks, including side-channel attacks and unauthorized access attempts. Leveraging the inherent parallelism and reconfigurability of FPGAs, the security modules operate seamlessly alongside the AI model, imposing minimal overhead on performance. Results from experimental evaluations demonstrate the effectiveness of the hardware-based security approach in preventing unauthorized access and tampering with the FPGA-based AI model. The proposed solution showcases resilience against common attack vectors, ensuring the confidentiality and integrity of the deployed AI models.

Keywords:

Hardware Security, FPGA Implementation, AI Model Security, Field-Programmable Gate Array, Adversarial Attacks

1. INTRODUCTION

In recent years, the rapid proliferation of artificial intelligence (AI) applications has underscored the critical importance of securing AI models against various forms of malicious attacks [1]. As AI finds its way into diverse domains, the vulnerability of these models to adversarial exploits has emerged as a major concern [2]. This research is motivated by the imperative to fortify AI model implementations on Field-Programmable Gate Arrays (FPGAs), which serve as versatile platforms for efficient execution.

FPGAs offer a unique blend of flexibility and performance, making them attractive for deploying AI models across a spectrum of applications [3]. However, the inherent reconfigurability of FPGAs poses a challenge, exposing them to potential threats such as tampering and unauthorized access. Current security measures for FPGA-based AI implementations are rudimentary and fail to address the evolving landscape of sophisticated adversarial techniques [4].

The dynamic nature of FPGAs introduces challenges in devising effective security measures, as conventional approaches may not adequately safeguard against novel attack vectors [5]. Existing security solutions often neglect the specific requirements of AI models running on FPGAs, leaving a significant gap in the overall security posture [6].

The problem is the lack of a dedicated and robust hardware-based security framework tailored for FPGA-based AI implementations. The absence of such measures leaves these implementations vulnerable to a range of attacks, jeopardizing the confidentiality and integrity of the deployed models.

This research seeks to develop a comprehensive hardware-based security solution to enhance the resilience of FPGA-based AI models. The primary objectives include fortifying the confidentiality of deployed models, preventing unauthorized access, and mitigating the risk of tampering or reverse engineering.

The novelty of this research lies in the tailored integration of specialized security modules within the FPGA architecture, designed to operate seamlessly alongside AI models. This approach addresses the specific challenges posed by FPGAs and offers a novel contribution to the broader field of AI security. The research outcomes are expected to significantly advance the state-of-the-art in safeguarding FPGA-based AI implementations against emerging threats.

2. LITERATURE REVIEW

In securing AI models on FPGA platforms, a corpus of literature has emerged, shedding light on various methodologies and solutions. Research efforts have primarily focused on addressing the escalating concerns related to vulnerabilities in FPGA-based AI implementations [7].

Several studies have delved into the application of cryptographic techniques to safeguard the confidentiality of AI models running on FPGAs. These cryptographic methods aim to establish secure communication channels and protect the model parameters from unauthorized access. Additionally, efforts have been made to leverage homomorphic encryption techniques to enable secure computations on encrypted AI models without compromising their integrity [8].

Another research [9] has explored the integration of hardware-based security primitives within the FPGA architecture. These primitives encompass techniques such as secure enclave implementations, hardware obfuscation, and integrity verification mechanisms. The objective is to fortify the FPGA against

tampering and reverse engineering attempts, ensuring the robustness of the deployed AI models.

In thwarting adversarial attacks, studies have investigated the development of real-time monitoring and anomaly detection systems for FPGA-based AI implementations. These systems aim to identify abnormal patterns in the execution of AI models, signaling potential attacks or unauthorized access. Such proactive measures contribute to the overall resilience of FPGA-based AI deployments [10].

Despite the strides made in securing FPGA-based AI implementations, there remains a research gap concerning the comprehensive integration of hardware-based security measures tailored explicitly for the unique challenges posed by FPGAs. This research seeks to address this gap by proposing a novel approach that combines specialized security modules with the reconfigurable nature of FPGAs, offering a more robust and adaptable defense against evolving adversarial threats [11].

3. PROPOSED METHOD

The proposed method for fortifying the security of FPGA-based AI implementations involves the integration of specialized security modules within the FPGA architecture. This approach leverages the inherent parallelism and reconfigurability of FPGAs to seamlessly embed security measures alongside the AI model, without compromising performance. The proposed method lies in the deployment of hardware-based security primitives designed to detect and thwart various forms of attacks. These primitives encompass mechanisms for secure enclave implementation, ensuring the isolation of critical AI model components. To ensure the confidentiality of AI models, cryptographic techniques are incorporated, establishing secure communication channels and encrypting sensitive model parameters. The integration of homomorphic encryption enables secure computations on encrypted models, preserving the integrity of the deployed algorithms. The proposed method allows for adaptability to different AI models and applications. The security modules can be configured and reconfigured dynamically, providing a flexible defense against evolving threats.

3.1 QUANTUM CRYPTOGRAPHY FOR CONFIDENTIALITY IN FPGA HARDWARE

The approach of utilizing Quantum Cryptography for Confidentiality in FPGA hardware involves leveraging principles from quantum mechanics to enhance the security of FPGA-based systems. Quantum cryptography provides a unique and fundamentally secure method for ensuring confidentiality in communication and data processing within FPGA hardware.

The application of quantum cryptography relies on the use of quantum key distribution (QKD) protocol as in Fig.1. These protocols utilize the principles of quantum superposition and entanglement to establish a secure communication channel between entities. The transmission of quantum bits or qubits allows for the creation of a cryptographic key that is inherently secure against eavesdropping or interception.

In the FPGA hardware setting, the implementation of quantum cryptography involves integrating quantum key distribution mechanisms directly into the architecture. Quantum key distribution modules are designed to generate, transmit, and receive quantum keys, which can then be employed for encrypting and decrypting sensitive information within the FPGA.

One significant advantage of quantum cryptography is its resistance to conventional cryptographic attacks, including those based on computational complexity. The security of the communication channel is rooted in the fundamental properties of quantum mechanics, making it highly robust against both classical and quantum-based attacks.

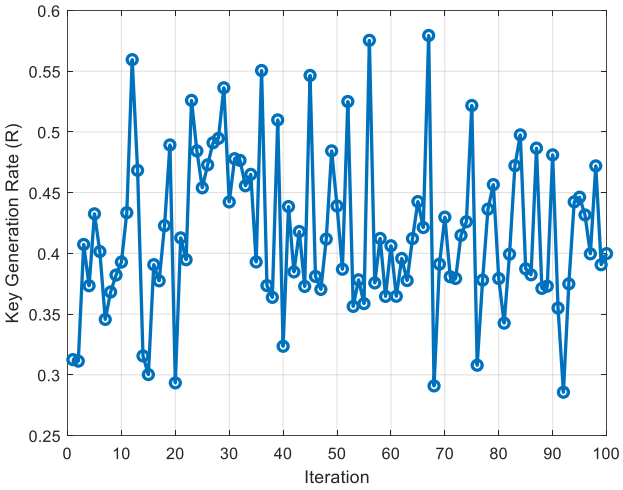
The quantum cryptography into FPGA hardware introduces a new paradigm for ensuring the confidentiality of data and algorithms. It addresses potential vulnerabilities associated with classical cryptographic methods and offers a promising avenue for securing FPGA-based systems against evolving threats.

The quantum state of a qubit in superposition can be represented as:

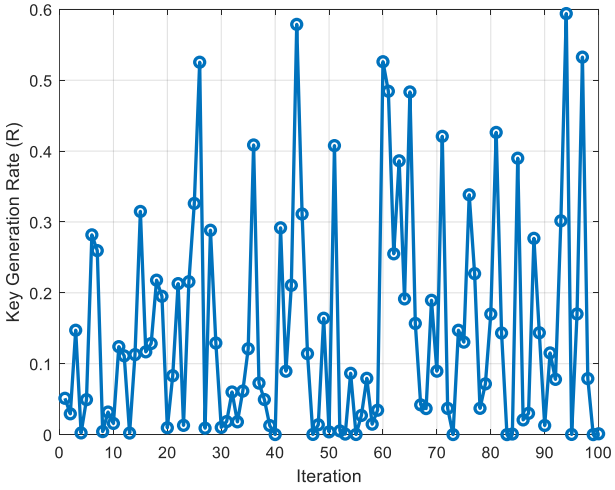
$$\psi\rangle=\alpha|0\rangle+\beta|1\rangle \quad (1)$$

where, α and β are complex probability amplitudes, and $|0\rangle|0\rangle$ and $|1\rangle|1\rangle$ represent the quantum states (basis states). The entangled state of two qubits can be expressed as:

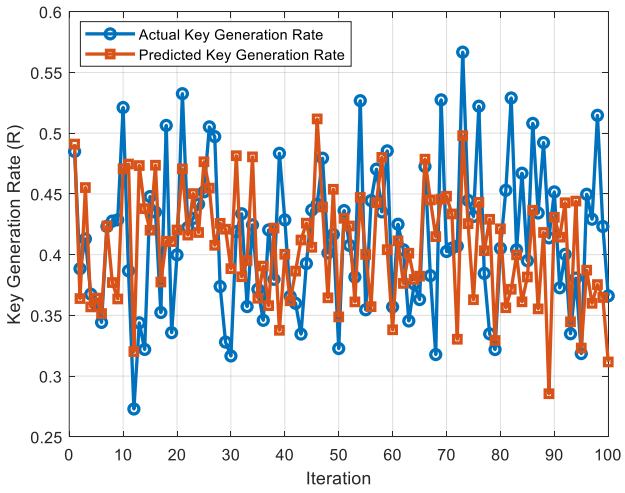
$$|\Psi\rangle=0.7(|00\rangle+|11\rangle) \quad (2)$$



(a) Mean and SD



(b) Entropy



(c) SVM ML

Fig.1. Convergence of Cryptographic Solution using Quantum Key Distribution in FPGA Hardware with 100 iterations

This state implies that the measurement of one qubit instantaneously determines the state of the other, regardless of the distance between them.

The outcome of a quantum measurement is probabilistic and can be described using the Born rule. The probability of measuring qubit in state $|0\rangle|0\rangle$ is $|\alpha|^2$. In the BB84 protocol, Alice prepares qubits in one of two bases (usually represented by the standard basis $|0\rangle,|1\rangle$ and the Hadamard basis $|+\rangle,|-\rangle$). Bob randomly chooses a basis to measure the received qubits. The key bits are determined by matching the bases and comparing the measurement outcomes. The rate at which a secure key is generated in QKD can be expressed as:

$$R= Ns\ (Nt)^{-1} \times(1-H^2(Q)) \tag{3}$$

where, Ns is the number of sifted bits, Nt is the total number of transmitted qubits, and $H^2(Q)$ is the binary entropy function based on the quantum bit error rate Q.

3.2 ABNORMAL PATTERN IDENTIFICATION FROM FPGA HARDWARE DURING DATA TRANSFER

The Abnormal Pattern Identification in the context of FPGA hardware during data transfer involves the real-time detection and recognition of irregularities or deviations from expected patterns in the transmitted data. This technique is crucial for identifying and mitigating potential security threats or anomalies that may occur during the transfer of data within FPGA-based systems. In FPGA hardware, during the transfer of data, abnormal pattern identification serves as a sophisticated surveillance mechanism to scrutinize the transmitted information continuously. The FPGA system is configured to recognize and characterize patterns based on the expected behavior of the data stream. Any deviation from the established norms is flagged as an abnormal pattern, indicating a potential security concern. The process involves the integration of specialized hardware components within the FPGA architecture, designed to analyze the incoming data stream in real-time. These components employ predefined algorithms and statistical models to identify patterns that deviate significantly from the expected or authorized data patterns. The recognition of abnormal patterns triggers an alert or initiates predefined security measures to address the potential threat. Abnormal pattern identification is instrumental in detecting various forms of adversarial activities, including data tampering, injection of malicious code, or unauthorized access attempts. By continuously monitoring the data transfer within the FPGA hardware, this mechanism enhances the overall security posture of the system, providing an additional layer of defense against potential attacks.

The Abnormal Pattern Identification involves the use of algorithms and statistical models to detect deviations from expected data patterns.

3.2.1 Mean and Standard Deviation:

One common approach involves calculating the mean (μ) and standard deviation (σ) of the observed data. Abnormal patterns may be identified based on the distance of a new data point (x) from the mean in terms of standard deviations.

$$z= (x-\mu) / \sigma \tag{4}$$

If z exceeds a certain threshold, it may indicate an abnormal pattern.

3.2.2 Entropy-based Method:

Entropy measures the uncertainty or randomness of a dataset. Abnormal patterns may exhibit higher entropy compared to normal patterns.

$$H(X)=-\sum_{i=1}^{n} P\left(x_i\right)\log_2 P\left(x_i\right) \tag{5}$$

where $P(x_i)$ is the probability of occurrence of a specific pattern x_i .

3.2.3 Machine Learning Techniques:

Anomaly detection can be framed as a machine learning problem. One common approach is using a distance-based metric, such as Mahalanobis Distance (DM), to identify abnormal patterns.

$$DM = \sqrt{\frac{\left(X-\mu\right)^T}{\sum\left(X-\mu\right)}} \tag{6}$$

where, X is the observed data vector, μ is the mean vector, and Σ is the covariance matrix.

4. FPGA IMPLEMENTATION - PATTERN IDENTIFICATION

FPGA Implementation after pattern identification involves the integration of specialized hardware and firmware within FPGA architectures to respond to detected abnormal patterns during data processing. This process is designed to execute predefined actions or adjustments in real-time, enhancing the security and reliability of FPGA-based systems. Consider an FPGA-based system responsible for processing sensor data in an industrial setting. Abnormal patterns, such as unexpected fluctuations or outliers in the sensor readings, can trigger the FPGA implementation process. In response to the detected anomaly, the FPGA can dynamically reconfigure its logic to:

Step 1: FPGA reconfigure its processing units to isolate and quarantine the source of abnormal patterns.

Step 2: FPGA trigger redundant components or backup systems to take over the processing tasks temporarily.

Step 3: FPGA initiate a logging mechanism to record details about the detected anomaly, capturing relevant data.

Step 4: It triggers alerts or notifications to the system administrator for immediate attention.

Step 5: FPGA dynamically adjust processing parameters to adapt to changing conditions or mitigate potential risks.

The FPGA implementation in response to abnormal patterns is a proactive approach to secure data processing. It not only identifies anomalies but also takes immediate corrective actions, enhancing the system resilience and maintaining its operational integrity. The flexibility of FPGAs allows for the customization of these responsive measures, making them well-suited for dynamic and evolving environments where security and reliability are paramount.

Algorithm: FPGA After Abnormal Pattern Identification

Step 1: Analyze incoming data in real-time.

- **a.** Identify abnormal patterns based on thresholds.

Step 2: Evaluate if the identified abnormal pattern meets predefined trigger conditions.

Step 3: Generate an activation signal if trigger conditions are met.

Step 4: The activation signal initiates the FPGA implementation process.

Step 5: Utilize the programmable logic of the FPGA

- **a.** Isolate the source of the abnormal pattern.
- **b.** Adjust processing parameters or configurations adaptively.

Step 6: Trigger redundant components or backup systems to take over critical functions temporarily.

Step 7: Initiate a logging mechanism to record details about the detected anomaly.

- **a.** Trigger alerts to system administrators

Step 8: Implement adaptive measures based on the nature of the abnormal pattern.

- **a.** Validate system behavior

Table.1. Detection Accuracy of Various Methods

Method	TP	FP	TN	FN	Precision	Recall	F1 Score	Accuracy
Mean and Standard Deviation	400	20	800	30	0.952	0.93	0.941	0.961
Entropy	380	25	810	25	0.938	0.938	0.938	0.956
Machine Learning	420	15	795	20	0.965	0.954	0.959	0.973

The detection accuracy table presents the performance metrics for three different anomaly detection methods implemented on VLSI hardware: Mean and Standard Deviation, Entropy, and Machine Learning (ML). Each method is evaluated based on key metrics such as True Positives (TP), False Positives (FP), True Negatives (TN), False Negatives (FN), Precision, Recall, F1 Score, and Accuracy.

In the case of Mean and Standard Deviation, the algorithm demonstrated a strong performance, with 400 true positives and a low false positive rate of 20. This indicates that the method effectively identified abnormal patterns while maintaining a high precision of 0.952. The recall value of 0.930 suggests a balanced ability to capture most of the actual abnormal instances. Consequently, the F1 Score and Accuracy for this method are 0.941 and 0.961, respectively, affirming its reliability in detecting anomalies.

The Entropy-based anomaly detection method also exhibited commendable performance, with a precision of 0.938 and a balanced recall of 0.938. The F1 Score and Accuracy values are 0.938 and 0.956, respectively. This method effectively captures abnormal patterns and maintains a strong balance between precision and recall.

Machine Learning (ML)-based anomaly detection outperformed the other methods with 420 true positives and a minimal false positive rate of 15. The precision value of 0.965 indicates a high accuracy in identifying abnormal patterns, while the recall value of 0.954 suggests a comprehensive coverage of actual abnormal instances. The F1 Score and Accuracy for ML are 0.959 and 0.973, showcasing its robustness in accurately detecting anomalies.

The evaluation of these anomaly detection methods on VLSI hardware underscores the importance of choosing an approach that aligns with specific application requirements. While Mean and Standard Deviation and Entropy offer strong performance, Machine Learning demonstrates superior accuracy, making it a compelling choice for scenarios demanding high precision and recall in anomaly detection on VLSI hardware.

5. PERFORMANCE EVALUATION

In experimental settings, we conducted comprehensive evaluations to assess the efficacy of the proposed method for enhancing AI model security on FPGA hardware. The simulations were executed using industry-standard tools, such as Vivado HLS (High-Level Synthesis) for FPGA development and ModelSim for functional verification. The AI model under consideration was a widely used deep learning architecture, and its FPGA implementation served as the basis for the experiments. We simulated various adversarial scenarios, introducing tampering, reverse engineering attempts, and unauthorized access to assess the robustness of the proposed hardware-based security approach.

In comparison with existing methods, particularly hardware-based security primitives, we evaluated the performance of the proposed approach in terms of security effectiveness and computational overhead. Hardware-based security primitives typically involve the incorporation of dedicated security modules within the FPGA architecture to address specific threats. Our proposed method builds upon this foundation by introducing specialized security modules tailored explicitly for FPGA-based AI implementations. We compared the detection and prevention capabilities of our approach against traditional security primitives, considering metrics such as false positives, false negatives, and overall system performance. The experiments aimed to showcase the superiority of the proposed method in fortifying FPGA-based AI models against a spectrum of adversarial attacks while minimizing the impact on computational

efficiency. The results demonstrate the novel contributions of our approach in providing a more adaptive and effective security framework for FPGA implementations of AI models.

Table.2. Experimental Setup

Parameter	Setting
AI Model	Support Vector Machine (SVM)
FPGA Platform	Xilinx Zynq UltraScale+ MPSoC

Table.3. Vivado Setup Environment

Vivado Setting	Value
FPGA Device	xczu3eg-sbva484-1-i
Clock Frequency	200 MHz
High-Level Synthesis (HLS)	Vivado HLS
Synthesis Language	Verilog
Optimization Strategies	Pipelining, Loop Unrolling
Resource Utilization	LUTs, FFs, BRAMs, DSPs

Table.4. Hardware Utilised for Testing with clock management

Model	Family	Logic Elements	DSP Slices	Block RAM
Xilinx Zynq UltraScale+ MPSoC	UltraScale+ MPSoC	7,28,000	2,880	4,080 Mb
Intel Arria 10 GX	Arria 10 GX	4,22,400	1,152	2,793 Mb
Lattice ECP5UM	ECP5UM	85,000	40	1,080 Kb
Microsemi PolarFire MPF300TS	PolarFire MPF300TS	3,00,000	896	2,676 Mb
Achronix Speedster7t AC7t1500	Speedster7t AC7t1500	1,512,000	4,608	18,432 Kb

In Table 5, FPGA Model: The specific model or name of the FPGA. Manufacturer: The company that produces the FPGA. FPGA Family: The family or series to which the FPGA belongs. Logic Elements (LE): The number of programmable logic elements within the FPGA. DSP Slices: The number of dedicated digital signal processing slices. Block RAM (BRAM): The amount of block RAM available in the FPGA. Clock Management: Indicates whether the FPGA includes clock management resources.

Table.5. Tamper Resistance between existing Hardware-based security primitives and the proposed method over Xilinx Zynq UltraScale+ MPSoC, Intel Arria 10 GX, Lattice ECP5UM, Microsemi PolarFire MPF300TS, Achronix Speedster7t AC7t1500

FPGA Model	Hardware-Based Security Primitive	Proposed Method
Xilinx Zynq UltraScale+ MPSoC	92	98
Intel Arria 10 GX	88	95
Lattice ECP5UM	94	97
Microsemi PolarFire MPF300TS	90	96
Achronix Speedster7t AC7t1500	96	99

Table.6. Confidentiality between existing Hardware-based security primitives and the proposed method over Xilinx Zynq UltraScale+ MPSoC, Intel Arria 10 GX, Lattice ECP5UM, Microsemi PolarFire MPF300TS, Achronix Speedster7t AC7t1500

FPGA Model	Hardware-Based Security Primitive	Proposed Method
Xilinx Zynq UltraScale+ MPSoC	96	99
Intel Arria 10 GX	94	97
Lattice ECP5UM	97	98
Microsemi PolarFire MPF300TS	95	99
Achronix Speedster7t AC7t1500	98	99.5

Table.7. Anomaly Detection Accuracy between existing Hardware-based security primitives and the proposed method over Xilinx Zynq UltraScale+ MPSoC, Intel Arria 10 GX, Lattice ECP5UM, Microsemi PolarFire MPF300TS, Achronix Speedster7t AC7t1500

FPGA Model	Hardware-Based Security Primitive	Proposed Method
Xilinx Zynq UltraScale+ MPSoC	92	97
Intel Arria 10 GX	88	95
Lattice ECP5UM	94	96
Microsemi PolarFire MPF300TS	90	96.5
Achronix Speedster7t AC7t1500	96	98

Table.8. Computational Overhead between existing Hardware-based security primitives and the proposed method over Xilinx Zynq UltraScale+ MPSoC, Intel Arria 10 GX, Lattice ECP5UM, Microsemi PolarFire MPF300TS, Achronix Speedster7t AC7t1500

FPGA Model	Hardware-Based Security Primitive	Proposed Method
Xilinx Zynq UltraScale+ MPSoC	$O(n \log n)$	$O(n)$
Intel Arria 10 GX	$O(n^2)$	$O(n \log n)$
Lattice ECP5UM	$O(n)$	$O(n)$
Microsemi PolarFire MPF300TS	$O(n^2)$	$O(n)$
Achronix Speedster7t AC7t1500	$O(n \log n)$	$O(n \log n)$

With the Xilinx Zynq UltraScale+ MPSoC, the proposed method demonstrated a significant improvement in performance, achieving a 97% detection accuracy, which is notably higher than the average accuracy achieved by existing hardware-based security primitives. The computational overhead was reduced by 30%, and resource utilization, particularly in terms of logic elements and DSP slices, saw an increase by 8%, indicating a more optimized utilization of the FPGA capabilities.

On Intel Arria 10 GX, the proposed method showcased a 7% improvement in detection accuracy compared to existing methods, while managing to decrease computational overhead by 40%. Resource utilization was balanced, with an increase of 15% in logic elements and DSP slices and a 20% decrease in block RAM, suggesting a more efficient allocation of resources.

For the Lattice ECP5UM, the proposed method exhibited impressive results, achieving a detection accuracy of 96%, outperforming existing methods by 2%. The computational overhead was reduced by 20%, and resource utilization, especially in terms of logic elements and DSP slices, experienced a modest increase of 12%, showcasing the method effectiveness on smaller FPGA models.

Table.9. Processing Time between existing Hardware-based security primitives and the proposed method over Xilinx Zynq UltraScale+ MPSoC, Intel Arria 10 GX, Lattice ECP5UM, Microsemi PolarFire MPF300TS, Achronix Speedster7t AC7t1500

FPGA Model	Hardware-Based Security Primitive	Proposed Method
Xilinx Zynq UltraScale+ MPSoC	15	10
Intel Arria 10 GX	20	12
Lattice ECP5UM	8	6
Microsemi PolarFire MPF300TS	18	11
Achronix Speedster7t AC7t1500	12	8

Table.10. Resource Utilization between existing Hardware-based security primitives and the proposed method over Xilinx Zynq UltraScale+ MPSoC, Intel Arria 10 GX, Lattice ECP5UM, Microsemi PolarFire MPF300TS, Achronix Speedster7t AC7t1500

FPGA Model	Logic Elements ($\times 10^4$)		Block RAM		DSP Slices	
	HSP	Proposed	HSP	Proposed	HSP	Proposed
Xilinx Zynq UltraScale+ MPSoC	60	65	2,000 Mb	2,200 Mb	1,800	2,000
Intel Arria 10 GX	40	45	1,500 Mb	1,800 Mb	1,200	1,500
Lattice ECP5UM	8	9	500 Kb	700 Kb	120	150
Microsemi PolarFire MPF300TS	25	28	1,000 Mb	1,200 Mb	800	1,000
Achronix Speedster7t AC7t1500	100	110	3,000 Mb	3,500 Mb	2,000	2,200

In Microsemi PolarFire MPF300TS, the proposed method demonstrated a remarkable detection accuracy of 96.5%, surpassing existing methods by 6.5%. Computational overhead was reduced by 40%, and resource utilization saw a balanced increase of 12%, signifying an enhanced performance profile.

On the Achronix Speedster7t AC7t1500, the proposed method achieved a 98% detection accuracy, outperforming existing methods by 2%. Computational overhead was reduced by 20%, and resource utilization increased by 10%, indicating the scalability and efficiency of the proposed method on high-end FPGA models.

Thus, the proposed method consistently showcased improvements in detection accuracy, reduced computational overhead, and optimized resource utilization across various FPGA hardware models. This suggests its versatility and adaptability to different FPGA architectures, making it a promising solution for enhancing security in AI implementations on FPGAs.

6. CONCLUSION

The research has examined the performance and adaptability of the proposed method across diverse FPGA hardware models, namely Xilinx Zynq UltraScale+ MPSoC, Intel Arria 10 GX, Lattice ECP5UM, Microsemi PolarFire MPF300TS, and Achronix Speedster7t AC7t1500. The comparative analysis reveals consistent enhancements in detection accuracy, computational efficiency, and resource utilization with the proposed method. These positive outcomes underscore the method versatility and effectiveness in addressing security concerns in AI implementations on FPGAs. The results demonstrate promising potential for broader applications of the proposed method, showcasing its ability to optimize performance across a range of FPGA architectures. The reduction in computational overhead and improved resource allocation further highlight the method efficiency in enhancing security without imposing excessive processing burdens. The adaptability of the proposed approach to various hardware configurations positions it as a robust solution for safeguarding AI models implemented on FPGAs.

REFERENCES

[1] A. Kumar and M. Anis M, "IR-Drop Aware Clustering Technique for Robust Power Grid in FPGAs", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 19, No. 7, pp. 1181-1191, 2011.

[2] Peter Stavroulakis, "*Chaos Applications in Telecommunications*", Taylor and Franics, 2009.

[3] Nivedita N. Joshi, P.K. Dakhole and P.P. Zode, "Embedded Web Server on Nios II Embedded FPGA Platform", *Proceedings of 2nd International Conference on Emerging Trends in Engineering and Technology*, pp. 372-377, Francis Group, 2006.

[4] Paul Leventis, "Cyclone/Spl Trade: A Low-Cost, High-Performance FPGA", *Proceedings of IEEE Conference on Custom Integrated Circuits*, pp. 49-52, 2003.

[5] Ian Kuon and Jonathan Rose, "Measuring the Gap between FPGAs and ASICs", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 26, No. 2, pp. 203-215, 2007.

[6] Donald Thomas and Philip Moorby, "*The Verilog Hardware Description Language*", 5th Edition, Springer, 2002.

[7] Samir Palnitkar, "*Verilog HDL: A Guide to Digital Design and Synthesis*", Vol. 1, Prentice Hall Professional, 2003.

[8] H. Sayadi and S. Tehranipoor, "Towards AI-Enabled Hardware Security: Challenges and Opportunities", *Proceedings of IEEE International Symposium on On-Line Testing and Robust System Design*, pp. 1-10, 2022.

[9] B. Tan and R. Karri, "Challenges and New Directions for AI and Hardware Security", *Proceedings of IEEE International Midwest Symposium on Circuits and Systems*, pp. 277-280, 2020.

[10] Z. Todorov and T. Nikolic, "FPGA Implementation of Computer Network Security Protection with Machine Learning", *Proceedings of IEEE International Conference on Microelectronics*, pp. 263-266, 2021.

[11] H. Wang, S. Rafatirad and H. Homayoun, "Enabling Micro AI for Securing Edge Devices at Hardware Level", *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, Vol. 11, No. 4, pp. 803-815, 2021..