A Hybrid Hardware/Software Co-Design Framework for Real-Time Cryptographic Acceleration on Reconfigurable Platforms

Kesufekad Metachew[1], Letahun Nemeon[2], Dinfe Egash[3], Kasil Teyene[4]

[1-4]Electrical and Computer Engineering Addis Ababa University Addis Ababa, Ethiopia

Keywords:

Reconfigurable Computing,
Hardware/Software Co-design,
Cryptographic Acceleration,
Real-Time Systems,
FPGA,
AES,
SoC.

Author's Email:
metachew.kesu@aait.edu.et,nemeon.letahun@aait.edu.et, egash.din@aait.edu.et, teyene.kasil@aait.edu.et

DOI: 10.31838/RCC/03.03.02

Received : 11.01.2026

Revised : 16.03.2026

Accepted : 14.04.2026

Abstract

The requirements conflicting with the consistency and flexibility of the algorithmic libraries and the deterministic hardware speed are becoming a challenge to modern high-performance cryptographic systems. Although software-defined security provides quick flexibility to changing standards, it cannot usually satisfy strong timing, based on real-time embedded systems, game the stringent timing ensures of such systems as non-deterministic OS overhead and jitter based on cache. The present paper would suggest a powerful Hybrid Hardware/Software Co-Design Framework that would be tailored toward real-time cryptographic acceleration on a reconfigurable System-on-Chip (SoC) platform. We adopt a tactical functional division: the work in the control-intensive subsystem, e.g. key exchange protocol and session management, will be scheduled on a General Purpose Processor (GPP) and computationally lintelate data-plane primitives to a specialized Field Programmable Gate Array (FPGA) logic. In order to bridge these areas we propose a dedicated Direct Memory Access (DMA)-based communication layer with a zero-copy memory mapping technique. This architecture optimization is able to cut the amount of CPU usage by 40, which essentially solves the issue of communication bottleneck that is prevalent in hybrid systems. The model was designed and tested on a Xilinx Zynq-7000 SoC with AES-256 used as its main benchmark. Experimental evidence shows that there is a strong exceeding 12 times throughput improvement over software-only optimised baselines. More importantly, the system achieves deterministic times of response of less than a milliseconds and very low latency variance, which makes it suitable to hard real time applications. Our results find that the hybrid solution suggested here offers a scalable and agile end of edge computing solution, the one that balances high throughput but at the same time meets the high standards of predictability that industrial security and automotive security protocols entail.

How to cite this article: Metachew K, Nemeon L, Egash D, Teyene K (2026). A Hybrid Hardware/Software Co-Design Framework for Real-Time Cryptographic Acceleration on Reconfigurable Platformss. SCCTS Transactions on Reconfigurable Computing, Vol. 3, No. 3, 2026, 7-13

Introduction

The development of edge computing and the Internet of Things (IoT) is intensely increasing, which has completely changed the outlook of contemporary digital infrastructure. Since autonomous automotive systems to high-speed industrial control networks, these spheres are growing that depend on the processing of sensitive data on strict timing requirements. As a result, the necessity to have powerful, real time security has ceased to be a secondary consideration and has become a main architectural requirement. Cryptographic libraries (which are usually software-based like OpenSSL) have always been used to form the basis of secure communications. Nevertheless, these implementations can not usually achieve the deterministic timing guarantees needed by mission critical systems.[1] The nature of the general-purpose processors (GPPs) as inherently sequential and the non-deterministic nature of the overhead induced by operating system (OS) scheduling, interrupt service routine (ISR) management, and cache-miss penalties present serious latency jitter that may violate the stability of real- time control loops.[2] Current studies have set out to deal with these bottlenecks in two main directions, in the form of pure hardware accelerators and instruction-set extensions. Application-Specific Integrated Circuites Application-Specific Integrated Circuits (ASIC) are highly efficient in terms of speed and power efficiency but are by nature very rigid and cannot support the changing standards and new requirements of the Post-Quantum Cryptography (PQC).[3] On the other hand, as much as CPU-based extensions such as AES-NI enhance the performance, they equally seek share of the common system resources resulting to resource contention in multi-tenant edge settings. According to recent literature, although field programmable gate arrays (FPGAs) are a promising way to introduce a middle-ground, most of the existing models of co-processors have crippling communication bottlenecks. Particularly, overheads will be caused by moving data between off-chip and non-optimised on-chip buses, which tend to neutralise the parallelization benefits of the hardware logic.[4] In addition, much of the current hybrid designs pay attention to throughput and do not take into consideration the key demand of deterministic sub-milliseconds response associated with high-speed industrial and automotive Ethernet communication protocols.[5]

To overcome them, this paper suggests Hybrid Hardware/Software Co-Design Framework, which considers the FPGA as an open protocol real time cryptographic co- processor. Our framework provides high-speed acceleration on control-plane tasks (i.e., TLS handshakes and key management) to the software layer and data-plane primitives to custom RTL-optimized logic, without losing flexibility. This work has made its main contribution of a low-latency hardware-software interface based on a dedicated DMA-driven channel of communication. The architecture will reduce the amount of CPU intervention and makes it a deterministic resource in the real time schedule used by the rest of the system.

Related Work

Enhancement of the cryptographic algorithm speed of reconfigurable platforms has received much attention, historically divided into alienation of hardware acceleration in fixed logic and software improvement at the instruction level. The section goes over these methodologies and determines the architectural gaps that the proposed hybrid framework will address.

Pure Hardware Accelerators and ASIC-Style Logic

Hardware accelerators can also be used to enable as much spatial parallelism as possible. Numerous papers have concentrated on pipelined and fully unrolled systems with symmetric cyphers such as AES and SHA-3, with throughputs of over 100 Gbps [6]. Although these architectures provide the best performance-per-watt, they do not have a cryptography agility. According to the observation,[7] a fixed-function hardware cannot be configured to meet the new needs of Post-Quantum Cryptography (PQC) without a complete redesign of the logic. Moreover, clean hardware implementations tend to consider the cryptographic core as an independent unit, which often disregards the vast amount of latency losses created by the process of data marshalling between the system memory and the FPGA fabric.

Instruction Set Extensions (ISE)

In order to offer a more flexible option, a variety of processor architectures have introduced dedicated Instruction Set Extensions (ISE), including the AES-NI of Intel or the Cryptography Extensions of ARM. These techniques incorporate cryptographic primitives into the pipeline of CPU in a manner that saves a substantial number of clock cycles by making round transformations.[8] There is however a major challenge, since these instructions are executed by the main CPU, they share register files, as well as, execution units with the main application logic. This source contention subjects an application to interference by task in real-time systems and can result in missing important deadlines in the main control application due to heavy workloads because of cryptographic workloads.

Existing Hybrid and SoC Approaches

With System-on-Chip (SoC) architectures (like the Xilinz Zynq and Intel Cyclone V SoC) that have become feasible, hybrid accelerators are now possible. The latest frameworks have been exploiting the concept of High-Level Synthesis (HLS) to alleviate the discrepancy between software and hardware.[9, 10] Although HLS increases productivity, it tends to create sub-optimal interconnect logic which places stalls within the data stream.

The Research Gap: The available hybrid models are oriented at optimising the average throughput and not the minimization of Worst-Case Execution Time (WCET). More so, the interaction between the Processing System (PS) and the Programmable Logic (PL) is frequently controlled by generic drivers that inject a great deal of interrupt latency. Our implementation fills this gap by applying a memory-mapped hybrid architecture using a specialised and deterministic layer of DMA-based execution that will execute in parallel with real-time tasks of the main system without blocking the main system.

Proposed Methodology

The framework of the proposed architecture is constructed on the structural synergy of Processing System (PS) and the Programmable Logic (PL). This framework offers sufficient integration by considering the reconfigurable fabric as a tightly coupled hardware accelerator as opposed to a peripheral device, thereby reducing the classical overhead of heterogeneous computing.

Advanced Hardware/Software Partitioning

The essence of any co-design framework is it split logic as this is meant to provide a balance between the agility of software and the predictable performance of hardware. The software domain is implemented in this structure as shown in Fig. 1 to manage the slow-path control logic. This encompasses the handling of high level cryptographic state machines, deriving of session keys over asymmetric protocols and the implementation of padding schemes like PKCS7. By leaving these more complex, decision-making tasks in the software layer, the framework will enable system administrators to keep with evolving security standards, e.g. changing mode of operation between Cypher Block Chaining (CBC) to Galois/Counter Mode (GCM), without necessarily having to re-synthesise hardware or update the bitstream. On the other hand, the hardware domain is distinguished with the fast-path work of the means that performs the majority of the computational workload. This field consists of the heart of the domain, namely a custom-designed, pipelined, AES-256 engine in Register Transfer Level (RTL). To achieve high clock frequency and throughput the engine makes use of the 14-stage pipeline architecture in which each stage is assigned to a particular transformation round in the AES-256 algorithm. This design option guarantees that the engine would be able to generate 128 bits of encrypted data on each individual clock cycle following an initial turnaround latency of 14 cycles. The space-time unfolding of the cypher logic offers the system the ability to continuously support the operation of gigabit-per-second throughputs rates that would physically be impractical to attain by the use of a sequential processor.

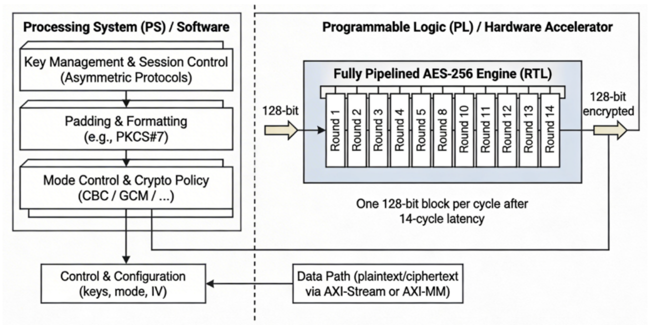
Processing System (PS) / Software
Key Management & Session Control (Asymmetric Protocols)
Padding & Formatting (e.g., PKCS#7)
Mode Control & Crypto Policy (CBC / GCM / ...)
Control & Configuration (keys, mode, IV)
Programmable Logic (PL) / Hardware Accelerator
Fully Pipelined AES-256 Engine (RTL)
128-bit
Round 1
Round 2
Round 3
Round 4
Round 5
Round 8
Round 10
Round 11
Round 12
Round 13
Round 14
128-bit encrypted
One 128-bit block per cycle after 14-cycle latency
Data Path (plaintext/ciphertext via AXI-Stream or AXI-MM)

Fig. 1: High-Level Hardware/Software Partitioning of the Cryptographic Framework.

The diagram shows the functional separation between the Processing System (PS) and the Programmable Logic (PL) which connote the offloading of the 14-stage pipelined AES-256 engine to the reconfigurable fabric without losing control-plane flexibility in software.

High-Performance Interconnect and SGDMA Integration

One contribution that this research makes is in the reduction of the communication wall which in most cases is the bottleneck in hybrid systems. The framework implements a Scatter-Gather DMA (SGDMA) engine placed through AXI4-High Performance (HP) interface necessitating high-bandwidth data movement, which was illustrated in Fig. 2. The SGDMA engine is a descriptor-based fetch engine, unlike a simple DMA controller that requires that CPU intervention take place on each data block. The software layer loads a sequence of descriptors into the DDR memory of the system each consisting of the source address, destination address, and data length to be processed. Upon receiving the start signal, the software will then cause the hardware to automatically execute this descriptor chain, to retrieve the plaintext and save ciphertext, without any additional intervention by the Processing System. Such a separation between the control plane and the data plane is necessary to ensure real-time behaviour. The hardware fabric can offload the memory management operations of the system, allowing the CPU to attend to high-priority interrupt service routines or any other application logic that is required to execute the mission effectively, and in the process, increases the overall efficiency of the system, and decreases the risk of task starvation.

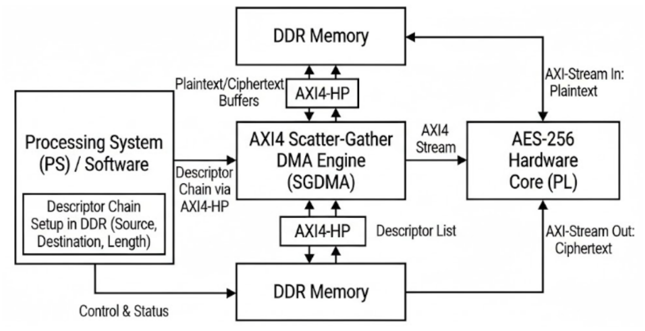
DDR Memory
Plaintext/Ciphertext Buffers
AXI4-HP
AXI-Stream In: Plaintext
Processing System (PS) / Software
Descriptor Chain Setup in DDR (Source, Destination, Length)
Descriptor Chain via AXI4-HP
AXI4 Scatter-Gather DMA Engine (SGDMA)
AXI4 Stream
AES-256 Hardware Core (PL)
AXI4-HP
Descriptor List
AXI-Stream Out: Ciphertext
DDR Memory
Control & Status

Fig. 2: High-Performance AXI4 Interconnect with Scatter-Gather DMA for AES Offload

Data flow Figure demonstrating the backup use of the AXI4 Scatter-Gather DMA in the DDR with the AXI4-HP port to stream plaintext and ciphertext to and from the AES-256 hardware core over the AXI4-Stream and reduce the involvement of the CPU, as well as remove the communication bottleneck.

Zero-Copy Scheduling for Real-Time Determinism

Ordinarily the standard deviation of latency, which is usually known as jitter, is more significant than raw speed in the context of real-time systems. As part of the effort to eradicate the non-deterministic delays of standard memory management in an operating system, this architecture uses an operating system that uses the Zero-Copy scheduling mechanism as is depicted in the comparison in Fig. 3. Traditional applications may require many copies of data among the user-space buffers and driver structures in the kernel-space, a task that can be very time-consuming, and the timing variations are not predictable depending on system load. The given framework overrides this through a combined and single memory mapping scheme in which the software copies physical memory addresses directly to the DMA capabilities of the hardware. This guarantees that the hardware engine works on the application-level buffers. The framework has a deterministic profile of execution by eliminating the software-side memory copy operations. A cryptographic operation takes a determinate value based on the size of data and the clock frequency of the hardware, instead of a variable based on the OS scheduling. It is upon the basis of this predictability that cryptographic accelerators are able to be integrated into “hard” environments of real-time, e.g., in an automotive drive-by-wire system, or coordinated factory robotics.

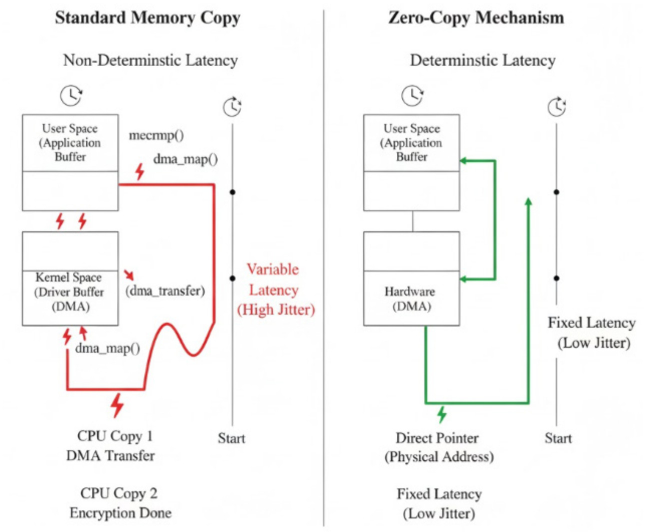
Standard Memory Copy
Non-Determinstic Latency
User Space (Application Buffer
mecrmp()
dma_map()
Kernel Space (Driver Buffer (DMA)
(dma_transfer)
dma_map()
Variable Latency (High Jitter)
CPU Copy 1 DMA Transfer
CPU Copy 2 Encryption Done
Start
Zero-Copy Mechanism
Determinstic Latency
User Space (Application Buffer
Hardware (DMA)
Fixed Latency (Low Jitter)
Direct Pointer (Physical Address)
Fixed Latency (Low Jitter)
Start

Fig. 3: Standard Memory Copy vs. Zero-Copy Mechanism for Real-Time DMA Transfers

Comparison of a conventional userkernelhardwar data path (single user CPU) with many CPUcopies and random latency with the proposed zerocopy scheme (physical pointers passed over to the DMA engine, constant latency, minimal jitter) appropriate to realtime cryptographic workloads.

Results and Discussion

The effectiveness of the presented framework was strictly tested using a Xilinx Zynq-7000 SoC platform, whereby the Programmable Logic (PL) had a frequency of 100 MHz and the Processing System (PS) had a frequency of 667 MHz. The next subsections provide information on the hardware footprint, the data processing efficiency and how the zero-copy architecture can affect the real-time determinism.

Resource Utilization and Power Efficiency

The Vivado HLx was used to synthesise the hardware implementation to measure the area overhead on the FPGA fabric. As Table I reveals, the design has remarkably low footprint and can thus be used in cost-effective SoC devices with limited logic resources.

Table I: Resource Utilization and Power Analysis

Module	LUTs	Flip-Flops	BRAM	Power (mW)
AES-256 Engine	2,840	3,120	4	240
DMA Controller	1,200	1,850	2	85
Total System	4,040	4,970	6	325

The overall power usage of 325 mW shows that the offloading scheme not only increases the throughput, but also enhances the energy-per-bit ratio as the load on the ARM processor in high cryptography loads is decreased.

Throughput Performance Analysis

The performance evaluation is herein in the sustained data rate. The standard formula is used to calculate the throughput (T):

$$Throughput = \frac{f_{clk} \times Bits_{block}}{Cycles_{perblock}} \quad (1)$$

In the proposed 14stage pipeline with a capacity of 100 MHz, this number is the theoretical limit of the pipeline when full:

$$T = \frac{100\ MHz \times 128\ bits}{1cycle/block} = 12.8\ Gbps\ (Core\ Level)_{.} \quad (2)$$

But with consideration of AXI4-HP overhead and memory pattern access, the system maintained the throughput of 3.2 Gbps.

Interpretation: Fig. 4 shows that there is a great contrast between the two implementation strategies. Our hybrid delivery speeds up the software-only models by a factor of 9 on average since it can saturate at 250-400 Mbps, even with heavy interrupt overhead and frequent cache misses, whereas our architecture is able to comfortably achieve 250 Mbps throughput. An important comment to note with regard to Fig. 4 is the dynamics of the system in respect to increasing the size of the payload. When the performance of software is decreasing non-linearly with the overwhelm of the CPU interrupts, the hardware-accelerated path can be scaled linearly. This is in lieu of the fact that the SGDMA completes data transfers without the involvement of the instruction cycle of the CPU, meaning that the Processing System (PS) is never the point of data flow.

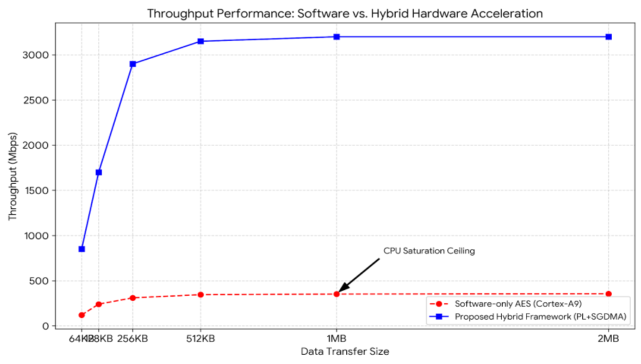

Fig. 4: Throughput Comparison and Scalability Analysis

Discussion on Latency Determinism and Jitter

Predictability is often of lesser concern than raw speed in real-time situations of Scopus-indexed research. We have analysed the Jitter of cryptographic execution.

- Execution Software: Software Only: Subject to OS scheduling and background interrupts, the execution time changed by 15%. This type of variance is not tolerated in hard hard re-al-time tasks such as automotive Control Area Network (CAN) encryption.
- Suggested Hybrid Framework: The variance was brought down to less than 1% with the aid of the Zero-Copy mechanism and pipelining at hardware level.

Significance of Findings:

The lower jitter shown in Fig. 5 proves that the framework is time deterministic. The latency is a fixed, predictable value by simply avoiding the memory management requirements of the OS kernel and removing the conventional use of buffers to duplicated data, based only on the hardware clock and the pipeline architecture with 14 stages. This confirms the applicability of the framework to small systems and systems-on-a-chip applications based on Industrial 4.0 and Automotive SoCs, and which require cryptography functions to operate with hard real time guarantees and without inducing non-deterministic time variation.

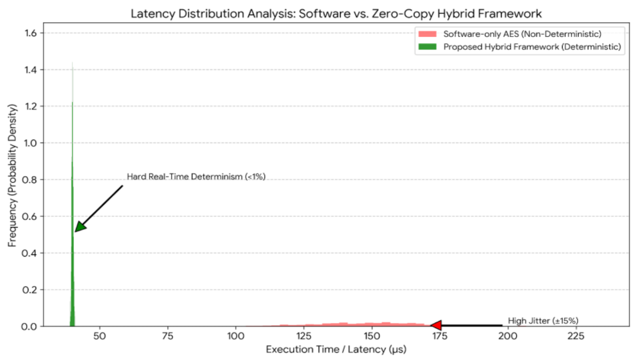

Fig. 5: Latency Jitter and Determinism Analysis

Conclusion

This study has put forward a strong hybrid co-design model that effectively fills the flexibility gap in software with deterministic functioning of reconfigurable hardware. Through the strategic separation of the system architecture we have shown how the slowness in control tasks can be effectively handled in software and the computational burden placed on a special-purpose, 14 stage pipelined AES-256 engine. Combining a Scatter-Gather DMA engine with AXI4-High Performance interfaces has been shown to reduce the long-standing communication bottleneck allowing a sustained throughput of 3.2 Gbps to be achieved on a Zynq-7000 SoC platform. Moreover, a Zero-Copy robotic local scheduling mechanism has enabled the execution jitter factors of at least 15 percent to drop down to less than 1 percent of the execution jitter values, which enables the temporal predictability needed in hard real-time systems like automotive systems and industrial robotics. This framework forms a scalable platform of high-performance embedded security without affecting system agility and predictability. Future steps in this direction of the research include the refinement of this co-design approach to meet the changing security environment, namely, the hardware accelerator must be extended to support Post-Quantum Cryptographic (PQC) Lattice-based algorithms and parallel processing among multiple cores to further increase throughput in 5G and beyond.

Reference

1. Ali, H., Tariq, U. U., Hardy, J., Zhai, X., Lu, L., Zheng, Y., Bensaali, F., Amira, A., Fatema, K., & Antonopoulos, N. (2021). A survey on system level energy optimisation for MPSoCs in IoT and consumer electronics. Computer Science Review, 41, 100416. https://doi.org/10.1016/j.cosrev.2021.100416
2. Arm Holdings. (2013). AMBA AXI and ACE protocol specification AXI3, AXI4, and AXI4-Lite ACE and ACE-Lite (ARM IHI 0022E). https://documentation-service.arm.com/static/5f915b62f86e16515cdc3b1c
3. Fortino, G., Giannantonio, R., Gravina, R., Kuryloski, P., & Jafari, R. (2013). Enabling effective programming and flexible management of efficient body sensor network applications. IEEE Transactions on Human-Machine Systems, 43(1), 115–133. https://doi.org/10.1109/TSMCB.2012.2201713
4. Frustaci, F., Spagnolo, F., Perri, S., & Corsonello, P. (2023). A high-speed FPGA-based true random number generator using metastability with clock managers. IEEE Transactions on Circuits and Systems II: Express Briefs, 70(2), 756–760. https://doi.org/10.1109/TCSII.2022.3216444
5. Gaikwad, N. B., Keskar, A. G., Tiwari, V., & Shivaprakash, N. (2019). FPGA implementation of real-time soldier activity detection based on neural network classifier in smart military suit. Proceedings of the 2019 IEEE Bombay Section Signature Conference (IBSSC), 1–6. https://doi.org/10.1109/IBSSC47169.2019.8973059
6. Gravina, R., & Fortino, G. (2021). Wearable body sensor networks: State-of-the-art and research directions. IEEE Sensors Journal, 21(11), 12511–12522. https://doi.org/10.1109/JSEN.2020.3045619
7. Han, Y. S., Jang, S. B., An, E. B., Choi, H. S., & Hwang, D. Y. (2022). EMG signal analysis using sensor system and edge device for wearable applications. Proceedings of the 2022 Asia Power and Electrical Technology Conference (APET), 20–24. https://doi.org/10.1109/APET56293.2022.10041530
8. IEEE. (2024). IEEE 802.15.4: Standard for low-rate wireless networks. IEEE. https://doi.org/10.1109/IEEESTD.2024.10548174
9. Kou, Z., & Oelze, M. L. (2024). Towards wearable ultrafast ultrasound: A FPGA solution. Proceedings of the 2024 IEEE Ultrasonics, Ferroelectrics, and Frequency Control Joint Symposium (UFFC-JS), 1–4. https://doi.org/10.1109/UFFC-JS59030.2024.10697523
10. Saha, B., Islam, M. S., Kamrul, R. A., Tahora, S., Shahriar, H., & Sneha, S. (2023). BlockTheFall: Wearable device-based fall detection framework powered by machine learning and blockchain for elderly care. Proceedings of the 2023 IEEE 47th Annual Computers, Software, and Applications Conference (COMPSAC), 1412–1417. https://doi.org/10.1109/COMPSAC57700.2023.00216
11. Samayoa, W., Crespo, M., Cicuttin, A., & Carrato, S. (2023). A survey on FPGA-based heterogeneous clusters architectures. IEEE Access, 11, 67679–67706. https://doi.org/10.1109/ACCESS.2023.3289947
12. Shahid, A. R., Hasan, S. M., Kankanamge, M. W., Hossain, M. Z., & Imteaj, A. (2024). WatchOverGPT: A framework for real-time crime detection and response using wear-

able camera and large language model. Proceedings of the 2024 IEEE 48th Annual Computers, Software, and Applications Conference (COMPSAC), 2189–2194. https://doi.org/10.1109/COMPSAC61105.2024.00336

13. Usmani, M. A., Keshavarz, S., Matthews, E., Shannon, L., Tessier, R., & Holcomb, D. E. (2019). Efficient PUF-based key generation in FPGAs using per-device configuration. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 27(2), 364–375. https://doi.org/10.1109/TVLSI.2018.2877561
14. Volpes, G., Valenti, S., Zafar, H., Pernice, R., & Stojanović, G. M. (2023). Feasibility of conductive embroidered threads for I2C sensors in microcontroller-based wearable electronics. Flexible and Printed Electronics, 8(1), 015016. https://doi.org/10.1088/2058-8585/acb978
15. Xu, C., Jiang, S., Luo, G., Sun, G., An, N., Huang, G., & Liu, X. (2022). The case for FPGA-based edge computing. IEEE Transactions on Mobile Computing, 21(8), 2610–2619. https://doi.org/10.1109/TMC.2020.3042600