Security of Infocommunication Systems and Internet of Things

2025 Vol 3, No 2

https://doi.org/10.31861/sisiot2025.2.02008

Received 02 August 2025; revised 17 November 2025; accepted 29 December 2025; published 30 December 2025

FPGA Platforms and Their Use in Edge Computing

Yurii Herman*

Yuriy Fedkovych Chernivtsi National University, Department of radioengineering and information security, Storozhynetska Str. 101, Chernivtsi, Ukraine

*Corresponding author (E-mail: herman.yurii@chnu.edu.ua)

ABSTRACT The article examines the role and future prospects of programmable logic devices (FPGAs) and system-on-chip FPGA (SoC FPGA) platforms within the edge computing paradigm. Particular attention is given to the combination of adaptability, fine-grained power consumption control, and high degrees of parallelism, which are critical characteristics for modern edge platforms. Additionally, the current state of FPGA adoption in practical edge scenarios is analyzed, ranging from video analytics in transportation systems to industrial vibration diagnostics and acceleration of telecommunications functions. Examples of both conventional FPGA-based solutions and hybrid SoC FPGA architectures are discussed, where programmable logic is tightly integrated with ARM-based processors to achieve balanced workload distribution between software and hardware components. It is demonstrated that such systems can significantly reduce processing latency, optimize energy consumption, and enable autonomous operation even under remote or unstable network conditions. The role of embedded operating systems is also examined, particularly in the context of SoC FPGA platforms, where Linux-based environments enable separation between control logic and hardware-accelerated data paths. The paper discusses how embedded operating systems influence system flexibility, software reuse, update mechanisms, and long-term maintainability of FPGA-based edge devices. In addition, the work addresses technical and organizational barriers that limit large-scale deployment of FPGA-based edge systems, including the lack of unified high-level synthesis toolchains, the steep learning curve associated with HDL-based design, and limited support for infrastructure-oriented workflows in FPGA-centric environments. Approaches for dynamic logic reconfiguration aimed at improving adaptability of local IoT systems are considered, along with challenges related to logic testing, system configuration, and scalability when adapting platforms to new application requirements. Modern development tools and frameworks for flexible system design, including cloud-based services and high-level programming environments such as Vitis HLS and Intel oneAPI, are also discussed in the context of reducing development complexity and accelerating design iterations.

KEYWORDS edge computing, FPGA, SoC FPGA, system flexibility, IoT.

I. INTRODUCTION

Edge computing shifts workload execution closer to data source, solving problems that arise from the core characteristics of Internet of Everything (IoE) and especially embedded domains. As a result of that shift - reliance on network and latency bottlenecks are becoming less critical, therefore allowing to focus more on specific responsibilities of selected devices. That approach is not novel but has become more broadly used due to the emergence of local processing and fast decision-making systems, which are highly adopted in embedded and specialized computing.

In this paradigm, hardware platforms that can do deterministic real-time processing and run on low power become very important. Traditional cloud-based architectures cause delays and extra bandwidth that cannot be avoided, which is not acceptable for tasks like industrial diagnostics, real-time video analytics, or fast decision making. Because of this, local processing architectures are relying more and more on different types of platforms instead of just general-purpose CPUs.

Field-Programmable Gate Arrays (FPGAs) and hybrid system-on-chip FPGA (SoC FPGA) platforms are different types of edge hardware because they allow implementation of task-specific parallelism directly in hardware layers. FPGA-based systems do not run instruction streams like CPUs and GPUs do. Instead, they use physically parallel logic structures, which let them have predictable latency and fine-grained control over power use. SoC FPGA platforms combine programmable logic with embedded processors. This makes it possible to separate high-level control software from data paths that are critical to performance.

When using FPGAs in edge environments, though, there are some design trade-offs that are not easy to circumvent [1]. Hardware acceleration can greatly lower processing latency and energy use, but it also makes the system more complicated, limits flexibility after deployment, and needs specifically grained development workflows. The lack of a single set of high-level tools and the steep learning curve that comes with hardware design are still practical barriers to widespread use.

As illustrated in Fig. 1, the classical edge computing architecture shifts the majority of computation to the local system, leaving only control or minimal interaction with external resources. While complete isolation is not mandatory, many practical deployments minimize external dependencies, except for essential operations such as system updates [2]. This article reviews the current state of FPGA- and SoC FPGA-based edge computing, focusing on architectural integration models, application domains, and system-level trade-offs.

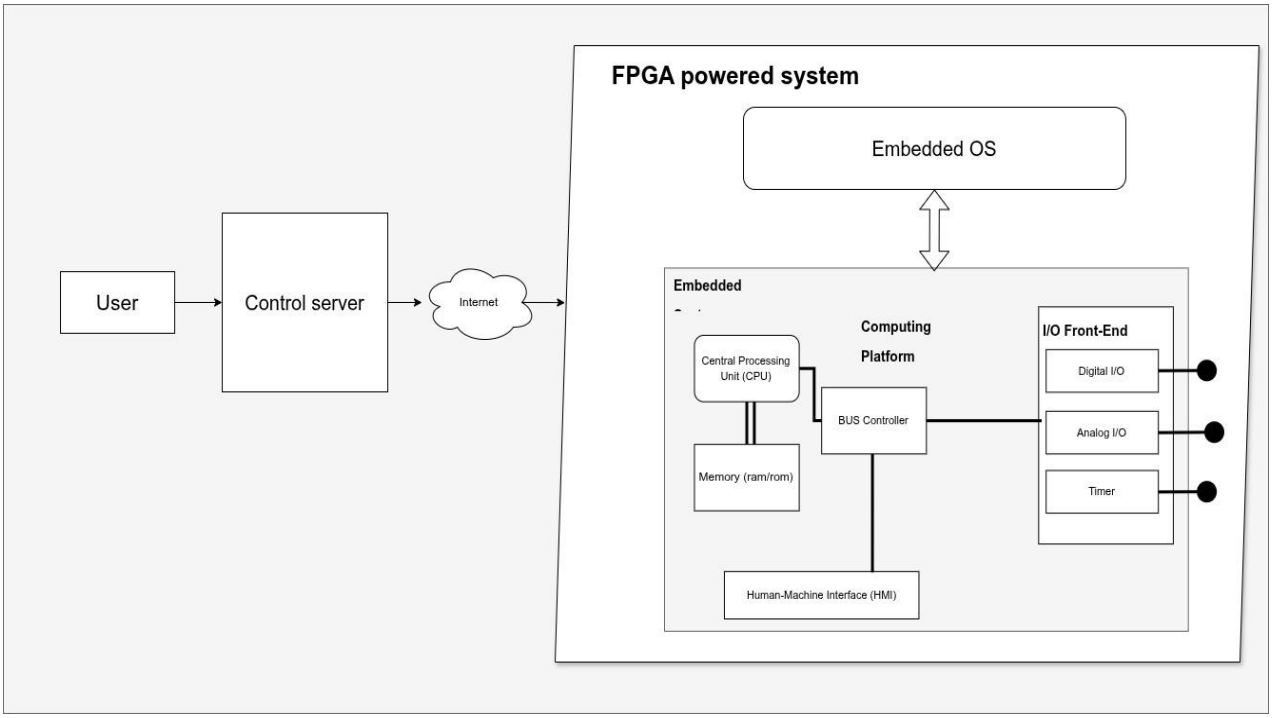


FIG. 1. Edge computing system architecture.

II. FPGA PLATFORM ADVANTAGES

FPGAs facilitate physical parallelism via autonomous logic blocks that function simultaneously, unlike CPUs and GPUs, which execute sequential instruction streams. This architectural characteristic is particularly vital for edge workloads requiring deterministic real-time behavior, where predictable latency sometimes supersedes peak throughput. Those features allow the shift of processing-critical stages into pipelines that are executed without scheduling overhead – therefore solving scheduling problem, that is prevalent on general purpose processors [3]. The ability to customize hardware logic for specific tasks is a significant advantage of FPGA-based systems. Even though software can be programmed to give CPUs and GPUs flexibility, they are still essentially general-purpose devices. FPGA can provide narrow execution pipelines that are specifically designed for those tasks, so highly specific logic like filtering or extraction is often done directly on FPGA [4].

Especially for simpler algorithms, getting rid of unnecessary abstraction layers makes it possible to implement them directly in hardware, which can speed up execution and use less energy [5]. Hybrid SoC FPGA platforms, which are often based on ARM architectures and combine programmable logic with embedded processors, make this technique better. This combination makes it possible to separate high-level control software and performance-critical data channels. FPGA-based edge devices can be easily added to existing systems because they come with standard operating systems and software ecosystems, so they do not need any extra computing power [6]. But this architectural flexibility comes at a cost: more complicated designs and tighter integration between hardware and software components. Another important part of edge deployments is how well they use energy. FPGA platforms let you control power use very precisely by enabling techniques like selective logic block activation, dynamic voltage and frequency scaling (DVFS) [7]. External memory access, often responsible for the majority of power consumption in embedded systems, can be significantly reduced by utilizing internal memory blocks (BRAM) and dedicated DSP units. Rather than centralizing computing in a singular high-frequency unit, several architectures allocate workloads among multiple slower parallel blocks to enhance power efficiency. A similar structure is shown in Fig. 2.

The benefits and limitations of FPGA-based edge systems are illustrated through practical applications. A road surface fault detection platform, based on AMD-Xilinx platform, reached frame-level latency of less than 50 ms by using an integrated CPU for advanced decision-making and FPGA logic for picture preprocessing [7]. However, changes to the recognition algorithms meant that the FPGA setup had to be recompiled and redeployed, which made the system less flexible. Comparable trade-offs are seen in industrial vibration analysis systems, where FPGA-based processing necessitates specialized development expertise that is rare among conventional SCADA engineers, yet ensures deterministic performance even at elevated temperatures. Alternative architectures demonstrate similar constraints in different forms. Case [8] describes an edge module based on a RISC-V microcontroller and Efinix Trion-based logic, where the FPGA processed the encryption protocol, and the main control remained in the MCU. This approach simplifies the system itself and avoids the need to integrate an ARM processor into the architecture. However, this system was highly specialized and created difficulties with integration into a large-scale infrastructure. FPGA platforms are particularly effective in edge scenarios that prioritize task-

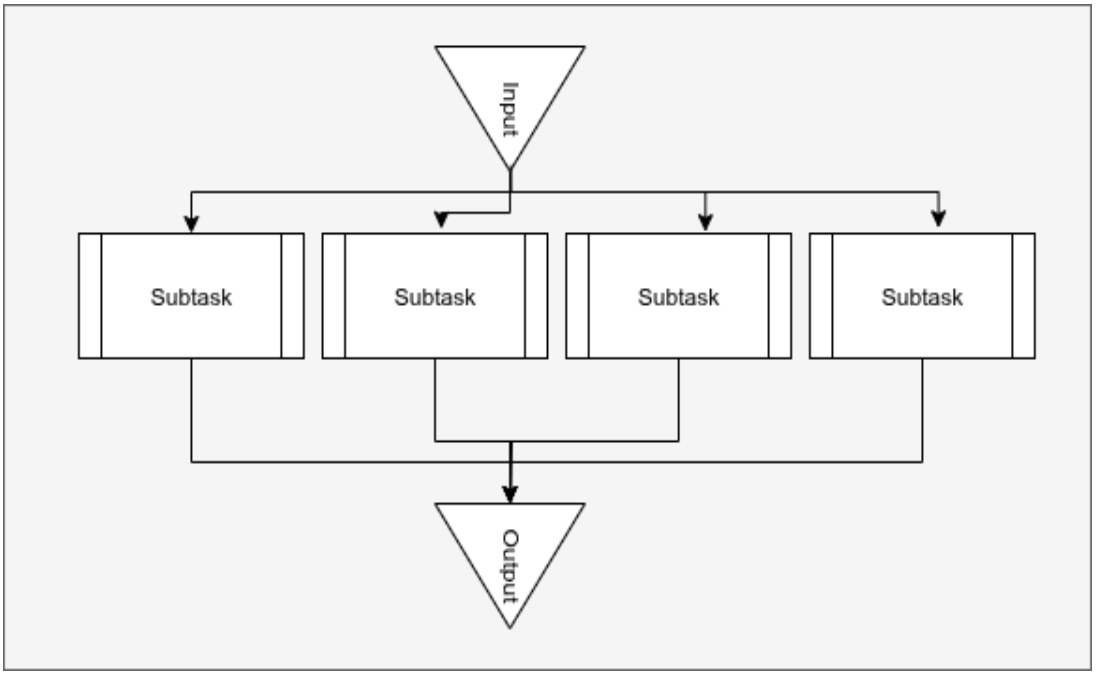


FIG. 2. Split processing diagram.

specific optimization, minimal latency, and deterministic execution. Those benefits need to be weighed against the need for specialized tools, more complicated development, and less flexibility after deployment [9]. So, FPGA-based solutions work best when workloads are clearly defined, and there are already set rules for how the system should behave and perform.

III. SPECIFICS OF INTEGRATION OF FPGA INTO EDGE-BASED ENVIRONMENTS

Most edge computing systems are built to reduce dependence on external resources and minimize outgoing traffic to outside infrastructure. This design paradigm has resulted in the development of various established integration models for implementing FPGA-based platforms in edge environments, each distinguished by distinct trade-offs among performance, flexibility, and development complexity. Using heterogeneous SoC platforms, like Xilinx Zynq or Intel SoC FPGA devices, is one of the universally common ways to achieve that aforementioned outcome. These devices combine an embedded processor and programmable logic on one chip, at the system design stage, that combination makes it possible to split workloads between the hard processor system (HPS) and the programmable logic (PL). These platforms are well suited for handling specific use cases, while keeping up with compute and power consumption levels, or building highly targeted systems that should cover time constrained operations. They are used a lot in IoT modules, smart cameras, and unmanned systems [10].

An alternative integration model relies on overlay architectures such as FINN, VTA, and DNNWeaver. These solutions use reusable accelerator templates on top of FPGA hardware to speed up development and make it easier to deploy AI workloads. Overlay-based designs usually give up some performance and resource efficiency, but they do speed up prototyping cycles and make it easier to adapt, which is often what is needed in edge AI applications where model iteration speed is more important than peak optimization [11].

Partial reconfiguration provides a flexible way to integrate the system into different environments, since it provides a mechanism to modify certain parts of the FPGA fabric while the program is running. This feature lets the same hardware platform change to meet different workloads or operational conditions, giving a way to update specific parts of the system in trusted execution environments, even allowing sensitive logic to be loaded only when it is required. But partial reconfiguration also complicates processing pipeline when it comes to latency, runtime verification, and secure bitstream management, specifically in systems where deterministic security guarantees are required and based on hardware logic.

FPGA-based edge platforms require a tailored workflow to handle data and organize workloads, no matter which integration model is chosen. Execution models that focus on dataflow often use tiling strategies, which break down computational tasks into smaller pieces that can be mapped onto limited FPGA resources with little or no interdependence. Memory management generally complicates resulting system due to need of balancing between SRAM, BRAM, caching (especially its hits and invalidation) and external DRAM in a way that should guarantee reduced access latency and avoid bottlenecks that could hang whole system. To make solving those tasks easier, high-level abstraction tools like HLS and domain-specific languages (DSLs) like Vitis HLS and OpenCL are used. These tools make it easy to quickly use standard computational patterns, but they often do not provide control over hardware structures in great detail. Those are a compromise between development speed and low-level optimization, especially evident in edge applications where performance is important, but iteration speed still critical for environmental adaptation. Use of simulation and profiling tools, like Intel VTune FPGA Edition and Xilinx

Vitis Analyzer, gives an option to handle those trade-offs, by balancing between simplification of code and actual system performance via providing a way for finding possible bottlenecks and timing problems before the platform is put into use.

Practical deployments demonstrate that FPGA-based integration models can achieve substantial gains in both performance and energy efficiency under well-defined conditions. For example, paper reports up to a 2.7x reduction in energy consumption for signal processing workloads using a soft-core processor (NIOS II) coupled with a lightweight FPGA accelerator [12]. Similarly, reports an optimized implementation of the YOLOv3-Tiny neural network on a Xilinx Zynq-7020 SoC FPGA platform, achieving a per-frame latency of approximately 4.8 ms with a power consumption of 2.55 W [13], highlighting the potential of FPGA-based inference for latency- and power-constrained edge systems.

However, these results also expose fundamental challenges associated with FPGA-based AI inference. The dataflow-oriented architectures used in many FPGA accelerators are tightly optimized for specific models, making frequent updates or architectural changes costly in terms of redevelopment and validation. General absence of standardized toolchains and complicated availability for abstraction layers makes it harder to port AI workloads between different FPGA platforms and makes a way for unexpected functional differences between those ports. There is currently no universally accepted industry standard for introduction FPGA platform into an edge-computing sphere. For each instance, it is critical to consider the general workload, data density, time constraints, and power availability for each system. SoC FPGA platforms became common choice when latency is very important, and connectivity is not always reliable, but due to their agility they provide many widely different ways to solve the same task. Overlay-based methods provide a better balance between performance and development effort.

Overall, the combination of SoC FPGA platforms with embedded operating systems represents a robust foundation for building flexible edge systems, provided that reconfiguration mechanisms are applied with appropriate attention to security, verification, and long-term maintainability. While incorporation of operating system at base provides an option for integration of hundreds of common packages, outsourcing general logic into userspace while keeping an ability to handle heavy tasks using FPGA parts.

IV. CONCLUSION

As a result of the analysis presented above, FPGA and SoC FPGA platforms can be positioned as a technically justified foundation for edge computing systems, particularly in scenarios that impose strict requirements on timing determinism, energy efficiency, and operational autonomy. The architectural integration models discussed in this work demonstrate how deterministic execution, low-latency processing, and fine-grained power control can be achieved without reliance on centralized computing resources.

The analysis also shows that there is no one integration model that is best for everyone. Heterogeneous SoC FPGA platforms combine programmable logic with general-purpose processors to create a powerful balance between flexibility and performance. However, this comes at the expense of increased design complexity and the need for careful hardware–software co-design. Overlay-based methods make it easier to develop and speed up the process of making changes, especially in edge AI situations. But they do this at the expense of using resources less efficiently and having less control over low-level optimizations. Support of the partial reconfiguration functionality makes the system even more flexible, but it also makes it harder to check, slows down reconfiguration, and makes it harder to handle secure bitstreams.

One major problem with current FPGA-based edge systems is that they are not very flexible when it comes to quickly changing AI workloads. Many FPGA accelerators use tightly coupled dataflow architectures that work best with fixed computational graphs. This makes it expensive to update models often because they have to be redeveloped and validated. Also, fragmentation across toolchains and abstraction layers makes it harder to move and reproduce AI workloads across different FPGA families, which makes it harder to deploy large-scale and maintain them over time.

From a security point of view, the increasing use of dynamic reconfiguration makes it easier for attackers to find new ways to attack and harder to verify. It is still an open research question on how to ensure trusted reconfiguration, protect bitstreams, and keep behavior deterministic during partial updates, especially edge systems used in safety-critical or hostile environments.

So, future research should focus on making FPGA-based AI inference more adaptable by using more flexible accelerator architectures, standardized intermediate representations, and better compiler toolchains. Also important are better ways to reconfigure systems in a safe and verifiable way, as well as ways to make it easier to design hardware and software together without losing energy efficiency or determinism.

AUTHOR CONTRIBUTIONS

Y.H. – conceptualization, methodology, investigation; writing (original draft preparation), writing (review and editing).

COMPETING INTERESTS

The author declares no competing interests.

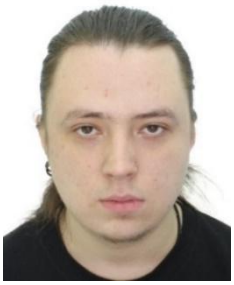
REFERENCES

[1] A. M. Sheikh *et al.*, “A survey on edge computing (EC) security challenges: classification, threats, and mitigation strategies,” *Future Internet*, vol. 17, no. 4, p. 175, 2025, doi: 10.3390/fi17040175.

[2] X X. Wang and W. Jia, “Optimizing edge AI: A comprehensive survey on data, model, and system strategies,” *arXiv:2501.03265*, 2025. [Online]. Available: https://doi.org/10.48550/arXiv.2501.03265.

[3] J.-Y. Zhan *et al.*, “FPGA-based acceleration for binary neural networks in edge computing,” *Journal of*

Electronic Science and Technology, vol. 21, p. 100204, 2023, doi: 10.1016/j.jnlest.2023.100204.
[4] P. Schulz and G. Sleahtitchi, "FPGA-based accelerator method for edge computing," *Preprints*, 2024, doi: 10.20944/preprints202405.0657.v1.
[5] O. Hryshchuk and S. Zagorodnyuk, "Managing energy consumption in FPGA-based edge computing systems with soft-core CPUs," *Journal of Edge Computing*, 2025, doi: 10.55056/jec.717.
[6] T. Wilson *et al.*, "Low-power FPGA design for edge computing systems," technical report, 2024. [Online]. Available: https://www.researchgate.net/publication/388753313_Low-Power_FPGA_Design_for_Edge_Computing_Systems.
[7] S. Chi, H. Lee, and J. Choi, "An edge computing system with AMD Xilinx FPGA AI platform for advanced driver assistance," *Sensors*, vol. 24, no. 10, p. 3098, 2024, doi: 10.3390/s24103098.
[8] Efinix, Inc., "Edge computing with FPGAs." [Online]. Available: https://www.efinixinc.com/blog/edge-computing-with-fpgas.html.
[9] S. Jiang *et al.*, "Accelerating mobile applications at the network edge with software-programmable FPGAs," in *Proc. IEEE INFOCOM*, Honolulu, HI, USA, 2018, pp. 55–62, doi: 10.1109/INFOCOM.2018.8485850.
[10] A. Boutros, N. Shanbhag, and D. Chen, "Field-programmable gate array architecture for deep learning: Survey and future directions," *arXiv:2404.10076*, 2024. [Online]. Available: https://arxiv.org/abs/2404.10076.
[11] W. Li, "Dataflow and tiling strategies in edge-AI FPGA accelerators: A comprehensive literature review," *arXiv:2505.08992*, 2025.
[12] Q. Shen, Y. Huang, and L. Wang, "T-Edge: Trusted heterogeneous edge computing," *arXiv:2503.16612*, 2024.
[13] R. Calì, L. Falaschetti, and G. Biagetti, "Optimized implementation of YOLOv3-Tiny for real-time image and video recognition on FPGA," *Electronics*, vol. 14, no. 20, p. 3993, 2025, doi: 10.3390/electronics14203993.



Yurii Herman

PhD student at Radio Engineering and Information Security Department of Yuriy Fedkovych Chernivtsi National University. Research field includes FPGA development, embedded systems and IoT.

ORCID ID: 0009-0003-2473-7365

Платформи FPGA та їх використання в кордонних обчисленнях

Юрій Герман*

Кафедра радіотехніки та інформаційної безпеки, Чернівецький національний університету імені Юрія Федьковича, Чернівці, Україна

*Автор-кореспондент (Електронна адреса: herman.yurii@chnu.edu.ua)

АНОТАЦІЯ У статті розглядається роль і перспективи застосування програмованих логікових інтегральних схем (FPGA) та платформ типу «система на кристалі» (SoC FPGA) у підході кордонних обчислень (Edge Computing). Особлива увага приділяється поєднанню адаптивності, точного контролю енергоспоживання та високого рівня паралелізму, які є критично важливими характеристиками сучасних edge-платформ. Проаналізовано поточний стан інтеграції FPGA у практичних сценаріях кордонних обчислень – від відеоаналітики в транспортних системах до промислової вібраційної діагностики та прискорення телекомунікаційних обчислень. Розглянуто приклади як традиційних FPGA-рішень, так і гібридних архітектур на базі SoC FPGA, у яких програмована логіка тісно інтегрована з процесорами на базі ARM для досягнення збалансованого розподілу навантаження між програмними та апаратними компонентами. Показано, що такі системи дають змогу зменшити затримку обробки, оптимізувати енергоспоживання та забезпечити автономну роботу навіть у віддалених або нестабільних мережевих умовах. Окрему увагу приділено ролі вбудованих операційних систем у контексті платформ SoC FPGA, де середовища на базі Linux забезпечують розмежування логіки керування та апаратно прискорених шляхів обробки даних. Обговорюється вплив вбудованих операційних систем на гнучкість архітектури, повторне використання програмного забезпечення, механізми оновлення та довгострокову підтримку периферійних пристроїв на базі FPGA. Також у роботі розглядаються технічні та організаційні бар'єри, що обмежують широкомасштабне впровадження FPGA-орієнтованих edge-систем, зокрема відсутність уніфікованих інструментів синтезу високого рівня, складність освоєння проєктування з використанням мов опису апаратури (HDL), а також обмежена підтримка інфраструктурно-орієнтованих робочих процесів. Проаналізовано підходи до динамічної реконфігурації FPGA-частини, що спрямовані на підвищення адаптивності локальних IoT-систем, а також проблеми тестування, конфігурації та масштабування при адаптації платформ до нових вимог функціонування. Сучасні інструменти та фреймворки для гнучкого проєктування систем, включаючи хмарні сервіси та високорівневі середовища розробки (Vitis HLS, Intel oneAPI), розглядаються в контексті зменшення складності розробки та прискорення ітерацій проєктування.

КЛЮЧОВІ СЛОВА кордонні обчислення, програмована логікова інтегральна схема, ПЛІС система-на-кристалі, гнучкі системи, Інтернет речей.