Proceedings of CONF-SEML 2026 Symposium: Computational Analysis and Modeling in Complex Intelligent Systems
DOI: 10.54254/2755-2721/2026.GU33833

Review of FPGA Hardware Acceleration and Co-optimisation of Hardware and Software for Artificial Intelligence

Suyao Wang

School of Electronic Information, Northwest University, Xi'an, China
sw23149@essex.ac.uk

Abstract. The rapid development of deep learning poses an unprecedented challenge to computing power. Constrained by the memory wall and energy efficiency bottlenecks, the traditional von Neumann architecture struggles to meet the diverse deployment requirements of AI applications in cloud and edge scenarios. Field Programmable Gate Array (FPGA) is suitable for accelerating neural network inference due to its unique advantages of high energy efficiency, low latency and hardware reconfigurability. This paper systematically reviews FPGA-based AI software-hardware co-optimisation technologies. Firstly, it sorts out the evolution process of the underlying architecture, from DSP-based multiply-add operations to on-chip memory hierarchy and Near-Data Processing (NDP). Secondly, it discusses the core strategies of software-hardware co-optimisation, including lightweight algorithms such as model quantization and structured pruning, as well as hardware design strategies such as pipelining and double buffering mapping. Then, it analyses the deployment in typical scenarios such as edge real-time detection and extreme environments. Finally, it discusses the memory bandwidth challenges and compilation toolchain barriers brought by large language models. In summary, based on the unique advantages of FPGA reconfigurability in adapting to algorithms, architectures and scenarios, this paper depicts a new paradigm of software-hardware collaboration of "software-defined hardware" based on FPGA, and prospects its great potential in achieving a system-level optimal balance among energy efficiency, flexibility and reliability in extreme scenarios.

Keywords: FPGA, AI, hardware architecture evolution, software-hardware co-design, edge scenario deployment

1. Introduction

Artificial intelligence is rapidly migrating from the cloud to the edge and extreme environments, placing higher requirements on computing systems in terms of power consumption, latency and reliability. Traditional GPUs can hardly meet the energy efficiency constraints of edge scenarios, while Application-Specific Integrated Circuits (ASICs) lack flexibility [1, 2]. FPGA has gradually become the preferred platform for deploying AI inference in key fields such as industry, vehicle-mounted and aerospace due to its reconfigurable logic, low-power parallel computing capability and strong real-time I/O characteristics. However, early implementations mostly relied on high-level

© 2026 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

DOI: 10.54254/2755-2721/2026.GU33833

synthesis tools or fixed acceleration modules, which often led to low resource utilisation, rigid scheduling strategies, and disconnection between algorithm and hardware design.

This paper systematically combs FPGA-based AI acceleration. Firstly, it analyses how the underlying architecture has shifted from relying on DSP to taking LUT as the core, and introduces near-data processing to alleviate storage bottlenecks. Then, it discusses how lightweight methods such as structured pruning and quantization work collaboratively with hardware optimisations such as loop unrolling and double buffering. Finally, it explains the adaptation logic in different scenarios through the real-time detection of YOLO at the edge and the radiation-resistant deployment of neural networks in space missions. The overall technical context of this paper is shown in Figure 1.

The main contributions of this paper are as follows: clarifying the internal correlation of the co-evolution of computing and storage architectures; proposing a closed-loop optimisation idea of "algorithmic regularity—hardware efficient mapping"; incorporating SWaP-C (Size, Weight, Power and Cost) and physical layer reliability constraints (especially radiation effects such as Single Event Upset (SEU)) into the co-design framework; and pointing out that Chiplet integration and unified software stack are the key directions to address memory and ecological bottlenecks in the era of large models.

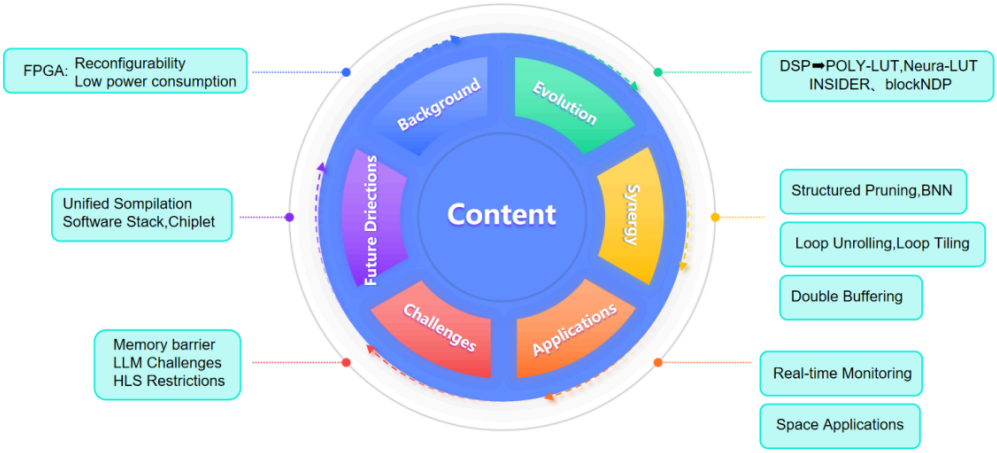

Figure 1. Research framework

2. Evolution of AI-oriented underlying FPGA hardware architecture

2.1. Transition from heavy reliance on DSP to LUT-centric architecture

As a multi-layer interconnected neural network, Deep Neural Network (DNN) has shown powerful feature extraction and regression capabilities in the fields of image recognition and natural language processing [3]. MAC operations are the main computing load of deep learning tasks, and traditional MAC operations are mainly performed by a large number of DSPs in FPGA. However, with the evolution of deep learning models towards lightweight and low complexity, the acceleration paradigm relying on FPGA gradually shows structural mismatch. Although DSP can implement MAC units in deep learning accelerators, its computing precision is usually higher than the demand of deep learning [4]. Even with superposition and pruning techniques, processing DNNs still consumes a large amount of valuable DSP resources. To solve this problem, a large number of studies have focused on the evolution from DSP to LUT with finer granularity and higher energy efficiency ratio. Compared with XNOR-based XNOR operations, LUT centred on non-linear

DOI: 10.54254/2755-2721/2026.GU33833

Boolean functions can further reduce the consumption of adder resources. With the emergence of technologies such as POLY-LUT and Neura-LUT, the industry has also launched hard-core tensor accelerators such as AMD/Xilinx AI Engine, which simplifies the deployment of low-precision networks with dedicated architectures and high-level programming, and further releases the advantages of FPGA in logic reconfigurability and performance optimisation.

2.2. On-chip memory hierarchy and near-data processing technology

However, under the trend of continuous expansion of deep learning model scale, even if the computing efficiency and throughput are significantly improved, the overall performance of the hardware will still be restricted if the problem of data synchronous supply cannot be solved. In traditional deep learning applications, BRAM in FPGA is mainly used to temporarily store computing operands and results. Nevertheless, high-throughput data transmission between memory and DSP or LUT will occupy a large amount of FPGA routing resources, leading to routing hotspots between memory and computing units and increasing energy consumption. In response to this problem, some studies have attempted to add Processing Elements (PE) to BRAM to realise on-chip computing functions. This technology provides a great deal of computing parallelism and reduces data movement [4]. For on-chip memory, other studies have proposed on-chip memory blocking and buffering to reduce access to external memory and thus solve the bandwidth problem. However, these technologies often encounter stuttering, area and overhead problems [5]. In recent years, Near-Data Processing (NDP), as a new architecture, is mainly characterised by offloading part or all data processing tasks to computing memory. This technology can not only reduce the burden on computing units but also lower data throughput. The University of California and Huawei Munich Open Lab have designed programming frameworks such as INSIDER and blockNDP respectively, providing a complete software stack and a consistent operating environment. The two are essentially different in the way of coupling logic and storage—INSIDER adopts a strongly coupled scheme of "hardware customisation + coarse-grained offloading", while blockNDP uses a weakly coupled scheme of "software definition + fine-grained block-level execution" [6]. Although they bring high software and hardware overheads, they reduce the development difficulty and alleviate the problems of lack of fixed standards and difficult task segmentation faced by NDP [6]. Although the ecology of NDP is still immature, it has broad development prospects as an effective means to solve the memory wall problem.

In summary, the continuous lightweight of deep learning models and the refinement of computing granularity have gradually weakened the demand for high-precision and high-power DSP units, thus providing space for the fine-grained reconfigurable logic centred on LUT to exert its advantages. This promotes the evolution of the underlying FPGA hardware architecture from DSP-centric to LUT-based, and lays a foundation for the implementation of new memory-computing collaborative paradigms such as NDP.

3. Software-hardware co-optimisation strategies

3.1. Algorithm-side lightweight

In extreme scenarios such as resource-constrained edge computing and space applications, if deep learning models (such as Transformer, ResNet) are directly deployed on FPGA platforms, their huge parameter scale and high memory demand will seriously exceed the on-chip BRAM capacity, resulting in ineffective operation or significantly reduced system lifetime [7]. Therefore, algorithm-

DOI: 10.54254/2755-2721/2026.GU33833

side pruning has become a prerequisite for efficient software-hardware collaboration. Although early unstructured pruning can compress the model volume, the generated sparse matrix lacks regularity, making it difficult for FPGA to maintain pipeline efficiency during processing, causing a large number of bubbles and complex memory addressing overheads, and leading to idle computing units and wasted routing resources [1, 8]. In contrast, structured pruning maintains the regularity of the matrix, enabling feature maps and weights to be efficiently blocked and mapped to on-chip memory units of FPGA, thus achieving highly parallel matrix operations and greatly improving actual hardware utilisation. In the field of FPGA-based deep learning object detection, Heller et al. adopted structured pruning technology to compress various YOLO networks, significantly reducing the model scale while effectively maintaining detection performance [9]. This trend has promoted the academic community to shift completely from unstructured methods pursuing extreme compression rate to structured pruning strategies friendly to FPGA pipelines. Table I compares structured pruning and unstructured pruning in different dimensions.

Table 1. Comparison between structured pruning and unstructured pruning

Index	Unstructured Pruning	Structured Pruning
Memory addressing overhead	High (indexing, random access)	Low (continuous access, no additional indexing)
Hardware parallelism	Low (irregular, difficult to quantify)	High (regular structure)
Accuracy loss	Small (with redundant weights removed precisely)	Large (Remove the entire set of parameters)

On this basis, quantization technology further promotes the evolution of models towards low-bit width: compressing the floating-point 32-bit (float32) precision to 8-bit integer (INT8), significantly reducing model storage occupation and data handling overheads. Furthermore, Binary Neural Network (BNN) compresses weights and activation values to ±1, simplifying complex multiplication operations into XOR and addition operations completely, which perfectly fits the LUT-based logic reconfigurable architecture in FPGA and greatly improves the hardware energy efficiency ratio [4]. For example, Reiter achieves 58.1 frames per second, which is 7.9 times faster than NVIDIA Tesla M60 GPU, with only 1/120 of the GPU's energy consumption, fully verifying the energy efficiency advantage of BNN in logic reconfiguration through LUT on FPGA platform [1]. To further break the limit of hardware efficiency, researchers have explored hierarchical compression technology (first pruning to reduce model scale and lower its regularity; then quantizing to reduce precision; finally binarizing to map operations directly to basic logic gates) so as to build a collaborative path from algorithm abstraction to physical implementation.

3.2. Hardware-side mapping acceleration

To give full play to the advantages brought by algorithm pruning, the hardware side must adopt a matching mapping acceleration strategy. The spatial computing architecture of FPGA determines that its performance depends not only on algorithm compression but also on how to efficiently schedule data streams to physical resources. Therefore, loop unrolling and loop tiling have become key methods. By unrolling loops in the dimensions of input and output channels, FPGA can convert serial computing into array processing with extremely high parallelism, solving the problem of low computing rate of traditional architectures [5]. Loop tiling further divides large-scale loops into small blocks adapted to on-chip memory, enabling weights and activation values to be reused on-chip, significantly reducing external memory access and alleviating the memory wall bottleneck

DOI: 10.54254/2755-2721/2026.GU33833

from the source. Zhang et al. decomposed convolution calculations into multiple parallel-processable small blocks through loop blocking strategy, significantly improving the data reuse rate of on-chip BRAM and keeping computing units continuously active at high throughput [9]. Other researchers have introduced cross-layer loop unrolling factors and loop blocking strategies to decompose convolution calculations into parallel-processable small blocks, and combined bit-serial LUT and DSP units to achieve a peak performance of up to 9.39 TOPS, significantly improving on-chip resource utilisation [5].

In addition, double buffering mechanism is the core strategy to hide cache latency and solve the mismatch between data supply and computing speed. Its core idea is to establish two physically isolated storage areas. When the computing unit processes data in the current buffer, the external data interface can simultaneously prefetch the next batch of data to another buffer, so as to achieve a dynamic balance between data supply and consumption and avoid idle computing units waiting for data [10]. In practical applications, this strategy is widely integrated into FPGA acceleration architectures. For example, early studies have integrated double buffering mechanism and parallel computing engines to achieve 600TFLOPS peak performance on Xilinx Alveo U280 platform, effectively masking memory access latency and greatly reducing computing latency and energy consumption [5].

In summary, the core of an efficient AI acceleration system for FPGA lies in the deep collaboration between continuous lightweight on the algorithm side and precise mapping on the hardware side. From structured pruning to binary neural networks, the model is not only gradually simplified but also improves hardware friendliness through regular sparse structure. More importantly, structured pruning ensures the continuous arrangement of remaining weights and activation values in memory by removing regular units such as entire rows, columns or channels. This data continuity provides a necessary prerequisite for hardware-side Single Instruction Multiple Data (SIMD) operations with bit-width expansion or the construction of large-stride deep pipelines. On this basis, optimisation technologies such as loop unrolling, loop tiling and double buffering further convert the regularity of algorithms into physical implementation with high parallelism and low latency. This closed-loop collaboration of "algorithm slimming" and "hardware efficiency enhancement" promotes the evolution of computing architecture from the traditional DSP-dependent paradigm to efficient logic reconfiguration centred on LUT, and finally realises a high-performance, low-power FPGA AI acceleration system.

4. Deployment analysis of typical edge and extreme scenarios

4.1. Real-time object detection in edge computing scenarios

In edge scenarios with strict power consumption resource (TDP) constraints, no reliance on cloud services and demand for real-time computing, the deployment of object detection models must balance algorithm efficiency and hardware adaptability. YOLO single-stage object detection algorithm is an ideal choice due to its regular structure and intensive computation, but its huge computation volume still poses challenges to hardware [11]. FPGA maps the regularised computation graph after structured pruning and quantization into a deep pipeline architecture through software-hardware collaboration, combined with loop unrolling, double buffering and efficient scheduling of on-chip BRAM, so that data streams advance continuously without frequent access to off-chip memory, thus achieving deterministic low latency under extremely low power consumption. For example, Heller et al. deployed the pruned YOLO network on Kria KV260,

DOI: 10.54254/2755-2721/2026.GU33833

achieving 15 FPS real-time detection with 3.5W power consumption, fully reflecting the value of system-level optimisation for edge scenarios [9].

4.2. High reliability and fault tolerance mechanism in extreme environments such as aerospace

In low-orbit satellite or deep space exploration missions, spaceborne intelligent systems need to process remote sensing data in real-time in strong radiation environments, and the reliability requirements are much higher than those of ground applications. Taking Synthetic Aperture Radar (SAR) imaging as an example, its signal has an extremely high dynamic range, and FPGA efficiently realises high-precision complex number operations through mixed precision, block floating point and bit growth management. However, cosmic rays easily induce Single Event Upset (SEU), leading to logic errors or even configuration failure, making general-purpose processors such as GPUs incompetent [12]. In contrast, FPGA can embed fault tolerance mechanisms such as Triple Modular Redundancy (TMR) and dynamic scrubbing directly into the hardware architecture by virtue of its reconfigurable characteristics, ensuring continuous and stable operation at the cost of limited resources. Among them, the execution frequency of scrubbing needs to be carefully weighed: too high refresh rate can improve system availability but significantly increase power consumption and bandwidth overhead; too low refresh rate fails to correct accumulated errors in time and endangers mission reliability. Some works have implemented radiation-resistant semantic segmentation accelerators on Zynq platform, significantly improving system robustness by enabling TMR on demand [13]. This indicates that the key of AI deployment for space applications is not peak computing power, but the collaborative design of algorithm lightweight and hardware fault tolerance, so as to achieve a system-level balance among accuracy, power consumption and survivability, and fully demonstrate the unique value of FPGA in extreme environments.

5. Current challenges and future development trends

5.1. Extreme memory bandwidth challenges brought by Large Language Models

Although the aforementioned software-hardware collaboration strategies have achieved remarkable results in lightweight models such as YOLO, FPGA deployment for Large Language Models (LLMs) still faces fundamental challenges [14]. As a typical memory-intensive task, the parameter scale of LLMs far exceeds the on-chip BRAM capacity, forcing the system to frequently access external DDR and causing a serious "memory wall" problem—computing units are often idle due to data handling latency, and energy efficiency drops sharply. Existing FPGA architectures are difficult to support their efficient inference. Future breakthroughs need to rely on 2.5D/3D integration schemes of Chiplet and HBM to alleviate the memory wall bottleneck of LLMs by raising the bandwidth upper limit of the FPGA roofline model.

5.2. Barriers of compilation toolchain and the future of unified software stack

The core challenge faced by FPGA in AI deployment currently lies in the lag of development paradigm: mainstream toolchains still rely on manual RTL design or limited HLS processes, requiring developers to be proficient in both deep learning model characteristics and underlying hardware timing, resource scheduling and other details, leading to difficulties in collaboration between algorithm and hardware teams and long iteration cycles. The contradiction between "energy efficiency advantage" and "development efficiency" seriously restricts the popularisation of FPGA

DOI: 10.54254/2755-2721/2026.GU33833

in the AI field. It is worth noting that Multi-Level Intermediate Representation (MLIR) is becoming the common foundation for AMD, Intel and other manufacturers to build scalable AI compilation stacks, bridging algorithms and hardware through multi-level abstraction, providing a feasible path to solve the above bottlenecks. In the future, it is urgent to build an end-to-end unified compilation software stack for AI workloads on this basis, automatically completing operator mapping, pipeline scheduling and storage optimisation through high-level abstraction, and transparently underlying hardware complexity to developers, so as to open a fast channel from algorithm prototype to efficient hardware implementation.

6. Conclusion

FPGA occupies an irreplaceable core position in the heterogeneous AI computing landscape pursuing extreme energy efficiency and low latency. This paper systematically combs the technical system of FPGA software-hardware co-optimisation for artificial intelligence. It clarifies the internal correlation of the co-evolution of computing and storage architectures from DSP dependence to LUT dominance and near-data processing, depicts the closed-loop optimisation path of algorithmic regularity and hardware efficient mapping, and creatively incorporates SWaP-C constraints and physical layer reliability requirements into a unified design framework, highlighting the unique value of FPGA in edge computing and extreme environments. Meanwhile, this paper points out that Chiplet integration and unified software stack are important technical means to solve the memory bottleneck and tool chain fragmentation in the era of large models.

Although this paper still has room for improvement in the breadth of literature, especially in the coverage of the latest industrial-grade compilation frameworks and cutting-edge chip architectures, it provides a clear technical evolution context for the development of software-hardware collaboration, and systematically analyses the deployment in edge scenarios and extreme situations. With the maturity of Chiplet micro-integration technology and the advent of reconfiguration technology based on autonomous agents, FPGA will surely redefine the physical and capability boundaries of silicon-based intelligent computing.

References

[1] Hozhabr, S.H. and Giorgi, R. (2025). A Survey on Real-Time Object Detection on FPGAs. IEEE Access. vol. 13, pp. 38195-38238. doi: 10.1109/ACCESS.2025.3544515.

[2] Hu, Y., Liu, Y. and Liu, Z. (2022). A Survey on Convolutional Neural Network Accelerators: GPU, FPGA and ASIC. 2022 14th International Conference on Computer Research and Development (ICCRD), Shenzhen, China. pp. 100-107. doi: 10.1109/ICCRD54409.2022.9730377.

[3] Guo Z. Y. (2025). A Survey on LUT-based Deep Neural Networks Implemented in FPGAs. arXiv: 2506.07367 [cs.AR]. https: //doi.org/10.48550/arXiv.2506.07367.

[4] Boutros, A., Arora, A. and Betz, V. (2024). Field-Programmable Gate Array Architecture for Deep Learning - Survey and Future Directions. arXiv: 2404.10076 [cs.AR]. https: //doi.org/10.48550/arXiv.2404.10076.

[5] Chatterjee, S., Ghosh, S., Ghosh, T. and Rahaman, H. (2026). Architectural Design and Performance Analysis of FPGA based AI Accelerators - A Comprehensive Review. arXiv: 2603.08740 [cs.AR]. https: //doi.org/10.48550/arXiv.2603.08740.

[6] Li, J.L., Liu, D., Chen, X.Z., Tan, Y.J. and Zeng, Z.Y. (2022). A survey of flash memory based near-data processing technology. Journal of Integration Technology, 11(3): 23-41. DOI: 10.12146/j.issn.2095-3135.20211019001.

[7] Meng Q.H. and Bian L.H. (2025). Research on FPGA-Based neural network hardware accelerators: A review. Journal of Signal Processing, 41(12): 1855-1873. DOI: 10.12466/xhcl.2025.12.001.

[8] Xu, B., Banerjee, A. and Gupta, S. (2025). Hardware Acceleration for Neural Networks: A Comprehensive Survey. arXiv: 2512.23914 [eess.SY]. https: //doi.org/10.48550/arXiv.2512.23914.

DOI: 10.54254/2755-2721/2026.GU33833

[9] Amin, R.A., Hasan, M., Wiese V. and Obermaisser, R. (2024). FPGA-Based Real-Time Object Detection and Classification System Using YOLO for Edge Computing. IEEE Access, vol. 12, pp. 73268-73278, doi: 10.1109/ACCESS.2024.3404623.

[10] Yunusoglu, A., Coskun, T., Vishwamith, H., Isik, M. and Dikmen I.C. (2026). A Reconfigurable Framework for AI-FPGA Agent Integration and Acceleration. arXiv: 2601.19263 [cs.AR]. DOI: 10.48550/arXiv.2601.19263.

[11] Dong J.D., Sang F.H., Guo Q.H, Chen, L. and Zheng, C.X. (2025). Review on Lightweight Research of Object Detection Algorithms Based on Deep Learning. Journal of Frontiers of Computer Science and Technology, 19(8): 2057-2084. DOI: 10.3778/j.issn.1673-9418.2503011.

[12] Bolchini, C., Cassano, L. Miele, A. (2023). Resilience of Deep Learning applications: a systematic literature review of analysis and hardening techniques. arXiv: 2309.16733 [cs.LG]. DOI: 10.48550/arXiv.2309.16733.

[13] Antunes, P. and Podobas, A. (2025). FPGA-Based Neural Network Accelerators for Space Applications: A Survey. arXiv: 2504.16173 [cs.AR]. https: //doi.org/10.48550/arXiv.2504.16173.

[14] Monteiro, C.T. and Guerrieri, A. (2026). Reducing Energy Footprint of LLM Inference Through FPGA-Based Heterogeneous Computing Platforms. Electronics, 15(5): 1052-1052. DOI: 10.3390/ELECTRONICS15051052.