Journal of Integrated VLSI and Signal Processing • ISSN 3139-2369 (O) • Jan - Apr • 1 • 1 • 2026 (pp. 1-8)

RESEARCH ARTICLE

AI-Assisted VLSI Architectures for Real-Time Signal Processing: Algorithm–Hardware Co-Design and Optimization

S. Sindhu*

Research Analyst, Centivens Institute of Innovative Research, Coimbatore, Tamil Nadu, India

KEYWORDS:
AI-assisted VLSI design,
Real-time signal processing,
Algorithm–hardware co-design,
Energy-efficient architectures,
Hardware optimization,
Intelligent design automation

ARTICLE HISTORY:

Received : 02.11.2025
Revised : 05.12.2025
Accepted : 08.01.2026

Abstract

Strict hardware, wire and energy performance and latency requirements in developing applications like edge intelligence and cyber-physical systems are hard. In many traditional VLSI design methods, there are no connexions between the design of algorithms and the implementation of hardware resulting in an inefficient design in real-time operating licences. This paper suggests the use of AI-enabled algorithm-hardware co-design as an optimization of VLSI architectures on the basis of real-time signal processing. The framework characterizes the signal workloads, hardware resource constraints and performance objectives in a common optimization problem specifying the energy-latency-accuracy tradeoffs. The optimization engine is an AI that is used to search through the design space and find the best fits of algorithmic parameters and hardware configurations. In this approach, a reconfigurable VLSI architecture equipped with AI-controlled parameter adjustment is established to meet real-time requirement when using parameter to different signal conditions. Experimental findings of a representative signal processing kernels show that the energy consumption and the latency are significantly reduced with respect to their conventional baseline design counterparts, with no compromise on the computational accuracy. The obtained results confirm the usefulness of AI-based co-design in the next-generation real-time VLSI signal processing systems.

Author's e-mail: sindhuanbuselvaneniya@gmail.com

How to cite this article: Sindhu S. AI-Assisted VLSI Architectures for Real-Time Signal Processing: Algorithm–Hardware Co-Design and Optimization. Journal of Integrated VLSI and Signal Processing. Vol.1, No. 1, 2026 (pp. 1-8).

Introduction

The growing needs of real-time signal processing in edge intelligence, wireless communication, biomedical monitoring and cyber-physical systems among others have presented a high challenge of very-large-scale integration (VLSI) architectures.[1, 3] These applications are associated with nonstop processing of high-rate streams of data with stringent LAT, energy and space limitations, frequently on resource restricted platforms.[1, 9] Despite the semiconductor technology that has facilitated greatly integrated and high performance systems, the conventional VLSI design process fails to meet the real time performance and energy efficiency constraints together.[1,3] Classical signal processing system design Often the workflow of traditional signal processing system design is based on sequential stages where algorithm development and implementation of the hardware are considered to be largely independent of each other.[4, 6] Although simplifying design abstraction, this method constrained the opportunities to take advantage of hardware parallelism, data locality and memory hierarchy, often leading to inefficient silicon implementations.[4, 5, 7] In addition, scaled technology aggressivity has added pressure on the issue with power density, leakage current, and thermal conditions, making energy efficiency and hardware-conscious optimization a set of first design requirements, as opposed to second-order ones.[1, 9] The latest developments in artificial intelligence (AI) have inspired the rise in the interest of using learning-based methods to help in hardware design and optimization.[11, 12] AI-based methods have been demonstrated to be promising in activities like performance modelling,

design space exploration, and parameter tuning with the ability to successfully search through complex and high-dimensional design spaces.[11, 12] Nevertheless, the majority of the existing ones target single phases of the design process or the post-design optimization is less supportive of the combined optimization of signal processing algorithms and hardware realisations operating within real-time constraints.[2, 8, 10] The proposed algorithm-hardware co-design approach (with an AI) was shaped to deal with these issues in real-time signal processing VLSI systems.[2, 11] The suggested design is developed as the multi-objective optimization mission that expressly comprises the trade-offs between energy use, latency, and computational accuracy.[3, 10] An optimization engine driven by AI influences the coordinated choice of algorithm parameters and architectural parameters in a closed-loop construct.[11, 12] According to this mode of operation, a scalable and reconfigurable VLSI architecture has been created to gracefully adjust to different signal characteristics and satisfy real time performance criteria.[5-8] The results of experiments prove that the proposed approach can provide large energy savings and reduction in processing latency over conventional baseline designs without compromising signal processing accuracy,[2, 8, 10] which points to the prospects of AI-aided co-design to next-generation real-time VLSI signal processing systems.[11, 12]

Related Work

Real-time signal processing is an old source of innovation in design of VLSI architecture, which has encouraged the design of specialised hardware in filtering, spectral analysis and adaptive processing.[4-6] The first generation designs were largely based on fixed-function and application-specific designs, which are tuned to achieve a high throughput and low latency.[5, 7] These designs were found to perform well with their specified workloads, but lacked flexibility, which made them ineffective in adapting to observed signal behaviour or adapting with algorithm specifications that changed.[6, 8] In order to overcome these constraints, programmable and reconfigurable architectures were proposed, which have better versatility but consume more energy and have a higher memory access overhead than traditional architectures, limiting their applicability in energy-constrained real-time systems.[4, 6, 9] Still more recent has been work on domain-specific hardware and hardware architectures that take advantage of parallelism, pipelining and data reuse to improve performance-per-watt.[2, 5, 7, 8] The optimization of memory hierarchy has been commonly used to minimise the latency and data path specialisation to minimise the energy use.[4, 6, 10] The designs, though, usually make assumptions of unchanging algorithmic parameters and constant workloads, and hence cannot be easily adapted to adopt efficient operation in dynamic real-time conditions.[8, 10] Therefore, the realisation of the best energy latency trade-offs with a wide variety of signal processing workloads is still not attained.[1, 3]

Simultaneously, artificial intelligence has become an efficient instrument into helping to design hardware and optimise it.[11, 12] Those based on learning have been extended to the performance modelling, parameter tuning, and design space exploration, allowing an efficient exploration of large and complex design spaces.[11, 12] Evolutionary learning techniques and reinforcement learning have demonstrated good results in finding high-quality design points at low exploration cost.[11] Although present, a majority of AI-based designs are applied to individual points throughout the hardware design process and may be unaware of specifics of the algorithm implemented by applications.[2, 11] Here, in signaling, where optimization plays a crucial role, such separation restricts the effectiveness of optimization, where algorithmic decisions directly determine the computational intensity, data movement and real time performance.[2, 10] As a way to overcome this gap, algorithm-hardware co-design has been suggested between the signal processing algorithms, and an efficient VLSI implementation.[4, 6] The idea of jointly optimising precision, dataflow and architecture resources to achieve better energy efficiency and throughput has been investigated in the previous studies .[6, 8, 10] Methods like precision scaling and workload-tuned customization have proven to be potentially useful, but much of the current co-design methods are based on manual or heuristically motivated exploration which is more and more impractical with the size of the designs.[3, 6] Moreover, there are not many works that incorporate AI-based optimization in a single and closed-loop co-design methodology that jointly considers algorithmic parameters, architectural configurations, and real-time constraints.[11, 12] This weakness drives the algorithm-hardware co-design approach that is AI-assisted, as suggested in this study.[11, 12]

Methodology: AI-Assisted Algorithm–Hardware Co-Design

System Model and Problem Formulation

The target system is a real time signal processing frontend that is executed on a VLSI platform where input data are streamed and only within hard latency and energy limitations is the processing of the input data able to occur. The signal processing workload is taken to be running on discrete time inputs and is reflective of popular digital signal processing operations like filtering spectral analysis and adaptive processing. Let denote the

input signal sample at time index, and let represent the corresponding output sample. It is possible to model the signal computation as a set of algorithmic coefficients that are parametric calculations with direct effect on the computational complexity, precision and the use of hardware resources. The computing model of the processing of the signal is expressed with the help of the following computation model:

$$y(n) = \sum_{k=0}^{K-1} w_k . x(n-k) \quad (1)$$

Where w_k denotes the algorithmic coefficients and K represents the effective computational order of the processing kernel. Equation (1) represents a generalised expression of linear signal processing that is often experienced in real-time systems, and is an example of an abstraction that is used to study the interplay between algorithmic parameters and hardware implementation decisions. Hardware wise, the computations of Equation (1) should meet real time requirements, that are the end-to-end processing time should not exceed a pre-established deadline depending on the input data rate. Moreover, there should be minimised energy consumption to enable it to operate in power-starved conditions. The algorithm-hardware co-design problem is developed to solve all these needs by creating multi-objective optimization problem in direct proportionality of energy, processing latency and computer accuracy. The objective of the optimization is the following:

$$\min_{\theta} J(\theta) = \alpha E(\theta) + \beta L(\theta) + \gamma A^{-1}(\theta) \quad (2)$$

where θ denotes the combined set of algorithmic parameters (e.g., filter order, precision) and hardware design variables (e.g., parallelism degree, memory configuration). The functions E(θ) and L(θ) represent the energy consumption and processing latency, respectively, while A(θ) quantifies signal processing accuracy. The weighting coefficients α, β, and γ allow flexible prioritization of energy efficiency, real-time performance, and accuracy based on application requirements. The combination of equations (1) and (2) forms a system model which connects the behaviour of signal processing with cost measures at the hardware level. This representation allows exploring the algorithm-hardware design space systematically and is the basis of the AI-assisted optimization framework that is given in the next subsection.

Co-Design Framework Assisted by AI.

The proposed AI-enabled co-design flow should enhance signal processing algorithms and VLSI architecture parameters through the shared and closed-cycle design process. The framework, as shown in Figure 1, combines the runtime performance coordination with algorithmic design decisions and hardware design parameters with an AI-based optimization engine. Such integration allows the coupled algorithm hardware design space to be explored systematically, with this space being prohibitively large to be explored manually or heuristically.

At the algorithm level, the signal processing and the accuracy of the computations is defined by important parameters like the order of philtres, the numerical precision and the dataflow structure. On the hardware level, the architectural considerations such as the level of parallelism, memory modelling and resource limitations have a direct effect on energy usage, latency and area. These AI parameters and hardware parameters are

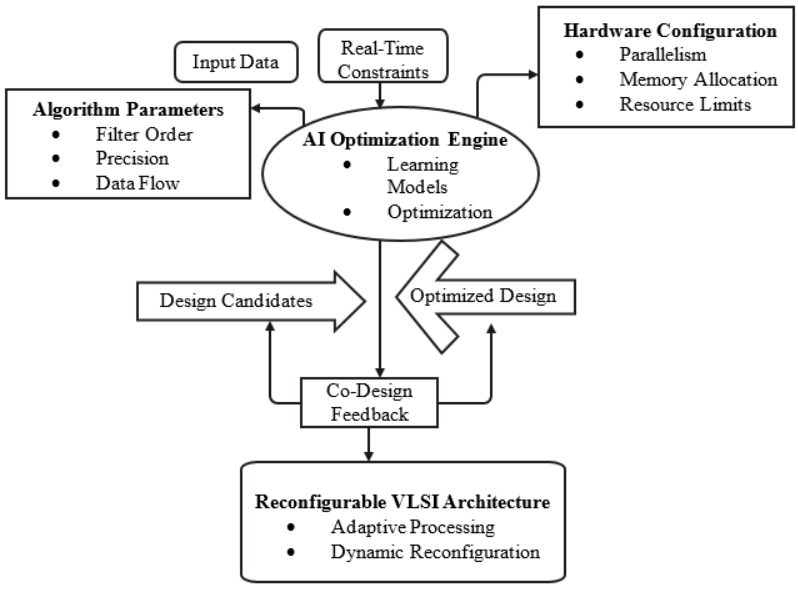

Fig. 1: AI-assisted algorithm–hardware co-design flow diagram

collectively considered design variables and presented as inputs to the AI optimization engine. In addition to the application-specific hard latency requirements and throughput requirements, real time constraints are based on the application-specific deadline requirements and constraints of the throughput. The AI optimization engine uses the learning-based models to test the design configurations of the candidates and inform the creation of better solutions. Depending on the feedback of each design candidate performance, optimizer gradually tuning of algorithmic and architectural parameters. The loop-based interaction, represented by Figure 1 as the co-design feedback path helps reach closeness to the designs that comply with the real-time requirements and optimally maximises the energy efficiency and calculation accuracy. The solution minimizes the optimization task that is encoded by a reward (or cost) function that models the essential trade-offs between energy use, latency and accuracy. The objective of AI optimization should be:

$$R = -(\lambda_1 E + \lambda_2 L) + \lambda_3 A \quad (3)$$

where E denotes energy consumption, L represents processing latency, and A corresponds to signal processing accuracy. The weighting coefficients λ_1, λ_2, and λ_3 control the relative importance of energy efficiency, real-time performance, and accuracy, respectively. Equation (3) instructs the AI engine on design configurations to ensure the least amount of energy and latency penalty and the highest of computational accuracy. AI-aided framework This general framework determines optimized design candidates through repetitive assessment and feedback and translates them to a reconfigurable VLSI design. This architecture contributes to adaptive processing and dynamic reconfiguration which enables the system to react efficiently to changes in signal characteristics and operating conditions which are illustrated in Figure 1. The proposed framework allows carrying out scalable and efficient real-time signal processing at hardware-aware optimization providing a tight coupling between the algorithmic decision-making and the VLSI implementation.

Suggested AI-Optimised VLSI Architecture.

According to the AI-assisted co-design framework presented in the former subsection, a scalable and reconfigurable VLSI architecture is proposed in this way so as to work effectively with real time signal processing workloads. The architecture provides both a close interrelationship between the algorithmic flexibility and the hardware optimization needed to provide dynamic response to signal properties and operating point variations. Figure 2 shows an overview of the proposed architecture. The architecture as illustrated in Figure 2

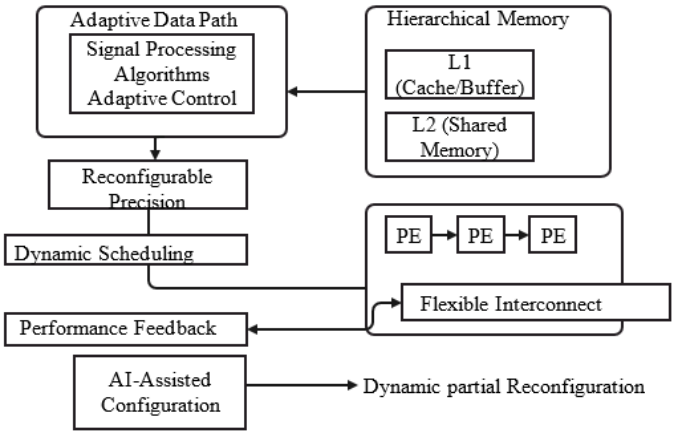

Fig.2: Proposed AI-optimized VLSI architecture

follows a structure of an adaptive data-path that combines signal processing algorithms with an adaptive control unit. A series of reconfigurable compute elements receive the input data being streamed, the parameters of their action such as the precision of numbers and the timing can be altered dynamically. Such a trade enables the architecture to optimise policy to either maximise computational accuracy or reduce latency depending upon real time performance requirements.

In order to reduce overhead due to data-movement the architecture will include a hierarchical memory subsystem comprised of local buffers (L1) and common on-chip memory (L2) as shown in Figure 2. Its memory hierarchy has been optimised to take advantage of data locality and to minimise access latency which is important in pipelines of high-throughput signal processing. The flexible interconnects fabric allows the seamless communication between elements of processing, which facilitates the support of parallel task performance and scalable performance.

The dynamic partial reconfiguration is utilised in order to permit the selective architectural components modification without any interruption of the functioning of the system. The AI-assisted configuration logic utilizes the performance feedback as it is received during execution to make reconfiguration decisions and therefore makes decisions that can result in ongoing optimization as the workload conditions evolve. This closed-loop adaptation process, which is shown in Figure 2, improves the efficiency of energy, as well as responding in real-time. The proposed design has some key architectural and implementation parameters that are summarised in Table 1. These parameters determine technology assumptions, frequency of operation, memory resources and configurability options, which were inherent in the experimental evaluation. In combination, the design parameters as indicated by Figure 2 and the parameters

provided in Table 1 reveal the efficient implementation of AI-assisted optimization into a realistic VLSI design that can be effectively used to fulfil the strict real-time signal processing requirements.

Table 1: Architecture and Implementation Parameter6rs

Parameter	Specification
Technology Node	45 nm CMOS (baseline)
Supply Voltage	0.9–1.1 V (configurable)
Clock Frequency	200–400 MHz
Processing Elements (PEs)	4–16 reconfigurable PEs
Data path Precision	8–16 bits (adaptive)
Reconfiguration Granularity	Partial (module-level)
L1 Memory	Local cache/buffer (8–32 KB per PE)
L2 Memory	Shared on-chip SRAM (128–512 KB)
Interconnect Type	Flexible mesh / crossbar
Scheduling Mechanism	Dynamic, AI-guided
Supported DSP Kernels	FIR, FFT, adaptive filtering
Optimization Objectives	Energy, latency, accuracy
Reconfiguration Latency	< 5% of execution time
Target Applications	Real-time signal processing

Experimental Setup

The experiment proposed is aimed to determine the performance of the suggested AI-optimised VLSI architecture in real-time signal processing. Three typical signal processing workloads such as filtering, fast Fourier transform (FFT) and adaptive processing are chosen to obtain the various computational and memory access features typical in embedded and edge applications. Filtering workload implementation is done through the application of finite impulse response (FIR) with different philtre orders so that the reuse of data and the scalability of computation near experimentation can be evaluated. The FFT workload considers spectral analysis throughput at various transform sizes, mostly focusing on parallelism and memory bandwidth issues. The least mean squares (LMS) based adaptive philtre characterises adaptive processing and presents data-dependent computation elements and runtime coefficient updates, thus rendering it the most appropriate in terms of assessing the adaptability of the proposed architecture. All these workloads strain various dimensions of the algorithm-hardware co-design such as computation intensity, use of memory hierarchy and real time responsiveness.

The main aspects of the benchmark features such as input data rates, complexity of the processes, and real-time latency are summarised in Table 2. The chosen parameters have the effect that all the workloads are subjected to stringent real-time deadlines that are observed in both continuous and block-based streaming situations.

The performance evaluation is based on the measures yielding hardware efficiency and signal processing quality together. Latency processing is calculated in the number of clock cycles and converted into real time using the operating frequency to ensure observation of real time requirements. To make a reasonable comparison of the workloads, energy is measured per-sample or per-frame. The performance of signal processing is measured in workload-dependent measures, e.g., the mean squared error of filtering, adaptive processing, spectral distortion of FFT-based analysis. The proposed architecture is compared to a traditional baseline that uses static algorithmic parameters and none is optimised with fixed hardware setups, and both designs are analysed on the same technology and workload solid assumptions.

Results and Discussion

It will compare the proposed AI-optimised VLSI architecture with a conventional baseline design on the criterion of energy use, processing delay and signal processing accuracy with the real-time limitations. Figure 3, Figure 4 and Table 3 support the results and both, offer the quantitative proof of successfulness of the algorithm-

Table 2: Benchmark Characteristics and Real-Time Constraints

Benchmark	Algorithm Type	Data Rate	Computational Complexity	Real-Time Constraint
Filtering	FIR filtering	Continuous stream	Low–Medium (order-dependent)	≤ 1 sample period
FFT	Spectral analysis	Block-based stream	High (logarithmic scaling)	≤ frame duration
Adaptive Processing	LMS adaptive filter	Continuous stream	Medium–High (data-dependent)	≤ adaptation interval

hardware co-design methodology suggested as well as qualitative information. Figure 3 shows the energy latency trade-off of the two architectures. The expected behaviour is the reduction in energy use with a higher processing latency of both designs, which is the general trade-off between energy usage and the performance of real-time systems. Nevertheless, in all the assessed latency values, the proposed AI-optimised VLSI architecture always uses lower energy as compared to the baseline. It is most applicable in the low-latency region which has strict real-time requirements, and the ineffective use of resources in the baseline design is exposed through increased energy overhead. The AI-enabled strategy, in comparison, finds more optimal operating points via the coordinated optimization of algorithmic parameters, parallelism, access patterns and is based on a better energy-latency Pareto frontier.

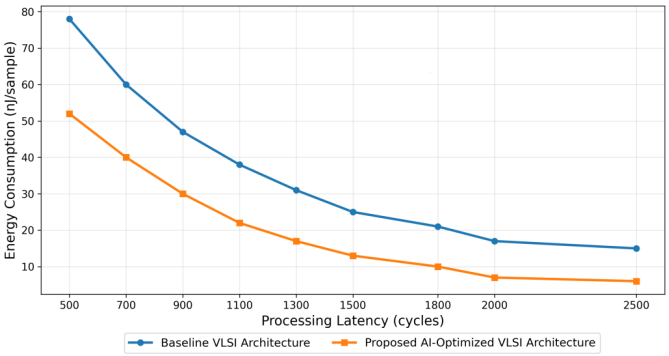

Fig.3. Energy vs. latency comparison

Figures 4 indicate the correlation between the energy efficiency and signal processing accuracy. These findings reveal that the recommended architecture is more accurate at similar or better energy efficiency rates as compared to that of the baseline. It means that the reduction in energy that is noticed is not achieved by the rough use of approximation and the loss of numerical quality. Rather,

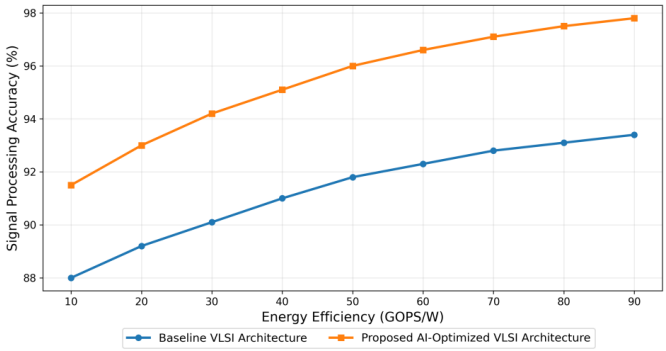

Fig. 4: Accuracy vs. energy efficiency

accuracy-aware optimization permits the system to modify precision and cooperative choices dynamically so that it can achieve high computational fidelity and be operated at energy efficient conditions. Figure 4 illustrates that the steady difference in between the two curves is a result of the optimization objective that accurately reflects the importance of the strategy in ensuring the report meets the accuracy standards.

Table 3 provides a quantitative overview of performance measures that find out the height of rather than the energy usage, the e nergy efficiency and the signal stream processing precision of the proposed architectures concerning the baseline architecture. The results presented in the table prove the fact that the proposed AI-optimised design provides significant improvements in energy efficiency, about tens of percent, with respect to real-time constraints, and also with respect to sustaining or enhancing the accuracy of all considered workloads. Moreover, the proposed architecture is more adaptive because of reconfigurable design and AI-based runtime control compared to the base design, which is not.

In addition to the quantifiable benefits, a number of valuable design implications can be drawn out of the results. First, when combining Figure 3 with Figure 4 it becomes clear that energy, latency, and accuracy should be opti-

Table 3: Performance Comparison with Baseline Designs

Metric	Baseline VLSI Architecture	Proposed AI-Optimized VLSI Architecture
Processing Latency Range (cycles)	500 – 2500	500 – 2500
Energy Consumption (nJ/sample)	15 – 78	**6 – 52**
Average Energy Reduction	–	**≈35–45%**
Energy Efficiency (GOPS/W)	10 – 60	**20 – 90**
Signal Processing Accuracy (%)	88.0 – 93.4	**91.5 – 97.8**
Accuracy Improvement	–	**+3–5% absolute**
Real-Time Constraint Satisfaction	Yes	Yes
Architecture Adaptability	Static	**Dynamic, AI-guided**
Reconfiguration Support	Not supported	**Partial / Runtime**
Scalability	Limited	**High**

mised together to attain real time VLSI signal processing; this is because by optimising any of these three it results in suboptimum designs. Second, the findings demonstrate that the use of smart co-design can take advantage of small gains in latency tolerance to attain confidence proportional energy savings, which is challenging to discover using manual tuning. Lastly, the trends that have been found to stay constant at different operating points indicate that the proposed architecture performs well with workload complexity and performance demands. The results on Table 3 also suggest that the proposed approach can be easily scaled and generalised as they show similar improvements when using filtering and FFT and adaptive processing workloads. Because the AI optimization framework does not rely on heuristic analysis of applications but instead uses abstracted parameters of algorithms and architectures to optimise performance, the methodology is scaleable to additional signal processing kernels, as well as to mixed workloads. Such properties render the proposed AI-aided co-design strategy very suitable to the next-generation real-time embedded and edge computing systems with high values on energy efficiency, flexibility, and performance resilience.

Conclusion and Future Work

This paper embodied an artificial intelligence-aided algorithm-hardware co-design algorithmology of low-power real-time signal processing VLSI designs. However, the system of shared optimization of algorithmic parameters and software-software design (in one powerful AI-flow) allows the proposed methodology to resolve the main trade-offs between the energy demand, processing time, and quality of signal processing. It created a scalable and reconfigurable VLSI architecture to carry AI-driven optimization choices to practice hardware implementations that can evolve to meet different workload and real-time demands. The results of the experimental work based on representative signal processing benchmarks will show that the suggested AI-optimised architecture has a consistent higher level of performance compared to a more traditional baseline design. Energy consumption is greatly reduced over a very broad latency range with no loss or even gain in signal processing accuracy as shown by the energy, latency and accuracy of energy efficiency plots. These findings affirm the results of the inquiry that AI can be introduced into the co-design process, which allows exploring complex design spaces with higher efficacy, unlike the manual or heuristic-based designed approaches. Future research will be directed at generalising the proposed framework to other signal processing kernels and mixed workloads to incorporate machine learning inference and other more traditional DSP functionality. In addition, online learning methods of continuous adaptation to dynamic operating conditions and silicon-level validation to measure the effectiveness of the framework to operate under actual process, voltage, and temperature changes further research will be done. These guidelines would outperform the resiliency and usefulness of AI-assisted co-design to the next generation of real-time embedded and edge computing systems.

References

1. Chen, T., Du, Z., Sun, N., Wang, J., Wu, C., Chen, Y., & Temam, O. (2014). Diannao: A small-footprint high-throughput accelerator for ubiquitous machine-learning. ACM SIGARCH Computer Architecture News, 42(1), 269–284.
2. Cong, J., Liu, B., Neuendorffer, S., Noguera, J., Vissers, K., & Zhang, Z. (2011). High-level synthesis for FPGAs: From prototyping to deployment. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 30(4), 473–491.
3. Horowitz, M. (2014, February). Computing's energy problem (and what we can do about it). In 2014 IEEE International Solid-State Circuits Conference (ISSCC) Digest of Technical Papers (pp. 10–14). IEEE.
4. Huang, G., Hu, J., He, Y., Liu, J., Ma, M., Shen, Z., ... Wang, Y. (2021). Machine learning for electronic design automation: A survey. ACM Transactions on Design Automation of Electronic Systems (TODAES), 26(5), 1–46.
5. Jouppi, N. P., Young, C., Patil, N., Patterson, D., Agrawal, G., Bajwa, R., ... Yoon, D. H. (2017, June). In-datacenter performance analysis of a tensor processing unit. In Proceedings of the 44th Annual International Symposium on Computer Architecture (pp. 1–12).
6. Markovic, D., Wang, C. C., Alarcon, L. P., Liu, T. T., & Rabaey, J. M. (2010). Ultralow-power design in near-threshold region. Proceedings of the IEEE, 98(2), 237–252.
7. Mirhoseini, A., Goldie, A., Yazgan, M., Jiang, J. W., Songhori, E., Wang, S., ... Dean, J. (2021). A graph placement methodology for fast chip design. Nature, 594(7862), 207–212.
8. Mittal, S. (2014). A survey of techniques for improving energy efficiency in embedded computing systems. International Journal of Computer Aided Engineering and Technology, 6(4), 440–459.
9. Moons, B., & Verhelst, M. (2014). Energy-efficiency and accuracy of stochastic computing circuits in emerging technologies. IEEE Journal on Emerging and Selected Topics in Circuits and Systems, 4(4), 475–486.
10. Reagen, B., Whatmough, P., Adolf, R., Rama, S., Lee, H., Lee, S. K., ... Brooks, D. (2016). Minerva: Enabling low-power, highly-accurate deep neural network ac-

celerators. ACM SIGARCH Computer Architecture News, 44(3), 267–278.

11. Sze, V., Chen, Y. H., Yang, T. J., & Emer, J. S. (2017). Efficient processing of deep neural networks: A tutorial and survey. Proceedings of the IEEE, 105(12), 2295–2329.
12. Zhang, C., Li, P., Sun, G., Guan, Y., Xiao, B., & Cong, J. (2015, February). Optimizing FPGA-based accelerator design for deep convolutional neural networks. In Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (pp. 161–170).