Journal of Integrated VLSI and Signal Processing • ISSN 3139-2369 (O) • Jan - Apr • 1 • 1 • 2026 (pp. 35-41)

RESEARCH ARTICLE

Real-Time and Low-Latency VLSI Implementations of Deep Learning-Based Signal Processing Algorithms

Ud. Chowdhury[1]*, Sohag Chakma[2]

[1, 2]Department of Electrical and Electronic Engineering, International Islamic University Chittagong, Chittagong 4318, Bangladesh

KEYWORDS:
VLSI architecture,
Deep learning,
Real-time signal processing,
Low latency,
Hardware acceleration

ARTICLE HISTORY:
Received : 15.11.2025
Revised : 20.12.2025
Accepted : 03.01.2026

Abstract

The recent establishment of deep learning (DL) methods in contemporary signal processing systems such as biomedical monitoring, radar sensing, speech enhancement, and image and multimedia processing has presented a high demand on real-time criteria, delay, throughput, and power use. Software-based realizations of deep neural networks are also notorious in embedded and edge development platforms, even though they can be used to achieve high accuracy inference, because of the erratic latency and high power consumption. This paper is an attempt to solve these problems by proposing a hardware-friendly VLSI implementation scheme of a deep learning-based signal processing algorithm to operate at real-time, and low-latency. The methodology proposed uses a hardware conscious deep neural network model directed into a parallel and pipelined architecture of VLSI that uses both data and task parallelism. An optimized processor elements that is combined with a streaming dataflow model, and localized memory buffering is employed to minimize end to end inference latency as well as overhead in accessing memory. The architecture will be in such a way that it allows running continuous signal processing in a very strict real-time setup and still be scalable in terms of resources. The suggested framework is tested and executed in an FPGA platform by applying post-synthesis and post-implementation analysis. It was also shown that experimental observations showed significant reduction of latency, throughput increase and reduced power consumption in contrast to a baseline non-optimized version of the same deep learning model. The results prove that hardware-software co-designing is a critical component to implement deep learning-based signal processing systems in real-time embedded systems. The suggested VLSI architecture will offer a feasible answer to the challenge of low-latency, low-energy-consuming intelligent signal processing in the next-generation edge and cyber-physical systems.

Author's e-mail: ud.chowdhury@gmail.com, soh_Chakma@gmail.com

How to cite this article: Chowdhury U, Chakma S. Real-Time and Low-Latency VLSI Implementations of Deep Learning-Based Signal Processing Algorithms. Journal of Integrated VLSI and Signal Processing. Vol.1, No. 1, 2026 (pp. 35-41).

Introduction

The recent deep learning (DL) has dramatically changed traditional signal processing paradigms by facilitating feature extraction, adaptive learning, and robust inference to a broad set of applications, practical ones being real-time biomedical signal processing, autonomous sensing, radar signal processing, speech processing, and multimedia processing.[1–3] DNNs have shown very high performance over conventional model based signal processing methods, especially in non stationary and complex environments. But these benefits of performance are at the cost of extremely high computational complexity and very high memory access performance demands that present serious challenges to support real-time operation on embedded and edge hardware. A large majority of state-of-the-art DL-based signal processing systems are executed on CPUs or GPUs, which are optimized towards high throughput but have non-deterministic latency and consume excessive amounts of power when required to work within real time constraints.[4, 5] These features render them unsuitable in latency sensitive applications like medical diagnostics, real-time monitoring and cyber-physical systems in which constrained response time and power efficiency are critical. Consequently, increased requirements exist to have specialized hardware acceleration solutions that are capable of providing guaranteed low latency and high

computational throughput. Implementations in VLSI have been suggested as considering one of the best solutions to these problems, to be able to take advantage of architectural parallelism, pipelining to deep extents, and workload-specific dataflow through the use of customized dataflow implementations.[6-8] Recent development suggested FPGA- and ASIC-based accelerators to deep learning, yet, much of the current literature conducts a throughput or batch-based deep learning inference in image classification applications.[9, 10] These designs fail to take into consideration end-to-end latency, streaming signal nature, and real-time conditions which are essential in the continuous signal-processing applications. Moreover, not all the reported architectures are reproducible because of the lack of experimental details and definitions of the comparison with the baseline.

In order to overcome these shortcomings, the framework of the real-time, low-latency VLSI implementation of deep learning-based signal processing algorithms is suggested in the current paper. The approach suggested puts an emphasis on hardware-algorithm co-design, streaming dataflow, latency-sensitive architectural design optimization that would allow the provision of continuous signal processing with pre-determined timing characteristics. The proposed architecture is designed with a focus on streaming signal workloads, as well as resource-constrained embedded systems in contrast to generic deep learning accelerator channels.

The key research findings contained in this paper are as follows:

- A machine-learned prediction system based on hardware limits and strict latency required of hardware-anticipated signal processing.
- Suppose precision equalizes precision (Alan Tolman, 1996): This is a parallel and pipelined VLSI architecture which exploits optimised dataflow and local buffering to minimise computational delay plus memory access latency.
- An experimental assessment of FPGA-based, following a repeatable implementation scenario, of latency, throughput, resource utilization, and power consumption relative to an unoptimized implementation of the same.

The rest of this paper will be structured in the following manner. Section 2 will conduct a review on associated literature on deep learning-based signal processing and VLSI acceleration methodologies. Section 3 provides the proposed methodology, which is system formulation, algorithm design, and VLSI architecture. Section 4 tells about the methodology and evaluation of the experiment. The discussion of the performance analysis and the results of the experiment are found in Section 5. Lastly, Section 6 presents the conclusion of the paper and future research directions.

Related Work

First VLSI applications of signal processing algorithms were based on fixed-function digital signal processing (DSP) blocks and application-specific integrated architectures specialized in well-defined operations like filtering, transforms and modulation.[11, 12] Although these designs were very efficient in classical signal processing tasks, they were not adaptive and scalable when faced with data-driven signal characteristics as well as non-linear ones. As deep learning rapidly developed, it has been served as the focus of serious research in hardware acceleration of deep neural networks (DNNs), especially convolutional neural networks (CNNs) and recurrent neural networks (RNNs). The accelerators can be FPGA based and ASIC based using the architectural techniques of loop unrolling, systolic array design, weight reuse, and reduced-precision arithmetic to enhance performance and energy efficiency.[3-6] These methods show significant improvements in throughput and energy consumption with respect to general purpose processors. Nevertheless, the majority of current deep learning accelerators are created with a focus on the maximum throughput or processing a batch of large data volumes, which are mostly image classification-based and computer vision-based workloads.[7, 8] With this kind of batch-oriented architecture there is usually buffering delays and non-deterministic latency, and no longer suitable to real-time signal processing systems where continuous data input is needed and there is a demand to meet time guarantees. In addition, most of the reported designs are based on optimization on an algorithm level with no consideration of end to end analyzing and streaming of dataflow needs.[9] A new line of research is now exploring the use of low-latency inference using pipeline optimization and memory-conscious scheduling.[10] However, such works typically do not have a co-design hardware/algorithms framework, and have low experimental reproducibility because of inadequate implementation information and baseline settings disclosure. Moreover, the trade-off of power and latency are not always considered in the context of realistic operating conditions of real-time.

This work, unlike the existing methods, is concerned with a latency-conscious VLSI architecture designed to support deep learning based streaming signal processing. With hardware-conscious algorithm development coupled with parallel architectural optimization and pipelined architectural optimization the proposed framework guarantees a deterministic low-latency operation at high energy usage and scaling. The methodology is a direct

solution to the limitation of previous studies and the strong linkage between deep learning algorithms and real-life VLSI implementation needs.

Methodology

System Overview and Problem Formulation

The proposed system will process continuous-time or discrete-time input signals and incur deep learning-based inferences on the inputs with rigid real-time constraints. We can denote the input signal as a discrete-time sequence x[n] and n is the index of samples. The signal will be divided into windows of the constant size N and create input vectors.

$$X_k = \{x[kN], x[kN+1],\ldots,x[kN+N-1]\} \quad (1)$$

which with the inference engine are sequentially executed. Fig. 1 shows an overview of the real time streaming architecture that has been proposed with windowing and buffering as well as VLSI inferred inference. The system aims at creating an output decision/estimate yk of each input window at the lowest end-to-end latency with inference accuracy remaining.

The overall minimization goal can be stated as the minimization of the overall inference latency.

$$L_{total} = L_{acq}+L_{comp}+L_{mem}+L_{out} \quad (2)$$

where L_{acq} denotes the signal acquisition and buffering delay, L_{comp} represents the computation latency of the deep learning model, L_{mem} accounts for memory access delay, and L_{out} corresponds to output generation latency. The network should meet real-time requirements, whereby the latency should be less than

$$L_{total} \leq L_{deadline} \quad (3)$$

where $L_{deadline}$ is application-specific and typically ranges from sub-millisecond to a few milliseconds.

Besides latency, the system has to maintain sustained throughput without loss of data as it runs under austere resource limits due to embedded systems. These limitations revolve around scarce logic resources, on board memory capacity as well as budget power. In order to meet these needs, the system uses a streaming processing model where signal samples are processed as they appear and avoiding processing the samples in batches meaning that no batch-level buffering is omitted and the timing behavior is deterministic. Fig. 1 illustrates that the entire system comprises of an input buffering station, a deep learning inference engine, which is VLSI based, and an output interface that are synchronized with a streaming dataflow.

Detail of schematic of suggested real-time signal processing pipe with input signal, x[n], chopped into

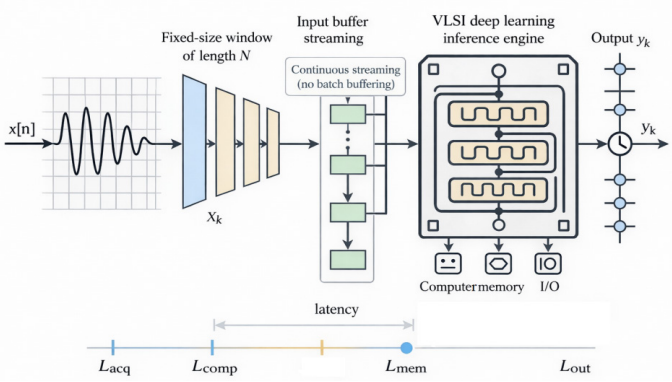

Fig. 1: Real-Time Streaming Deep Learning Inference Architecture Implemented in VLSI

fixed-size windows of length N and streamed through an input buffer to a VLSI-based deep learning inference engine. The system operates without batch buffering to ensure deterministic timing, producing output decisions y_k. The overall end-to-end latency is decomposed into acquisition (L_{acq}), computation (L_{comp}), memory access (L_{mem}), and output (L_{out}) components. Pipelined processing elements or neural layers that are used in implementing hardware are represented by stacked modules within the inference engine.

Deep Learning Algorithm Design

The deep learning architecture used in this paper is hardware efficient. The model involves convolutional layers then the lightweight fully connected layers, which is ideal in the task of temporal and spatial processing of signals. For a given convolutional layer l, the output feature map $Y^{(l)}$ is computed as

$$Y^{(l)} = f\left(\sum_{c=1}^{C} X_c^{(l-1)} * W_c^{(l)} + b^{(l)}\right) \quad (4)$$

where $X_c^{(l-1)}$ represents the input feature map of channel c, $W_c^{(l)}$ denotes the convolution kernel, $b^{(l)}$ is the bias term, $*$ denotes convolution, and $f(\cdot)$ is a nonlinear activation function.

The network is created with fixed kernel sizes and homogeneous layer structure with an aim of minimizing hardware complexity and enhancing execution efficiency, and providing the flexibility of regular hardware mapping. Quantization of weights and activations to fixed-point representations is used with reduced-precision arithmetic, which can withstand multiplier and memory demands much lower and with inference accuracies that can be sufficiently good. Moreover, optimization by layer is done so as to balance the computational load across layers, and no one layer is made a bottleneck of the processing pipeline in terms of latency. The training of model models

is conducted originally offline in a typical deep learning framework and the model parameters are exported in an exportable hardware-compatible format. The parameters are kept on-chip memory when they are provided or they are loaded through external memory by using specialised buffering echoing to the access point to reduce access fidelity.

Proposed VLSI Architecture

The proposed VLSI architecture is dedicated to the implementation of the hardware-aware deep learning model with low-latency and real-time execution. The Fig. 2 shows an overview of the proposed pipelined processing element (PE) array, hierarchy memory organization and streaming dataflow. It consists of a system of processing elements (PEs) that can process fixed-point multiplyaccumulate (MAC) operations and whereby this data structure forms the compute core of nutritional convolutional and fully connected layers. Production of individual PE in the tth clock cycle is treatable as:

$$y(t) = \sum_{i=1}^{K} w_i . x_i(t) + b \qquad (5)$$

where $x_i(t)$ and w_i represent the input data and corresponding weights, respectively, and b denotes the bias term.

The architectures will have deep pipelining across the stages of computation to provide0 high throughput and low latency so that multiple input samples can be processed by different stages of the process simultaneously. The exploitation of parallelism is through the distribution of calculations between multiple PEs in parallel to allow processing of independent channels or feature maps as illustrated in Fig. 2. The significant reduction of critical path delay and increment of effective processing rate are achieved through this parallel and pipe-lined execution. The hierarchical memory architecture is chosen so as to reduce the memory access latency and power usage. The local buffers are incorporated near the PEs where they are used to store commonly used data like intermediate feature maps and feature weights, and this limits the use of external memory. Information transfer between processing steps is in the form of a streaming dataflow, such that input samples get processed as they arrive not were they need to be stored in batches, which is in line with the architecture shown in Fig. 2. It is meant to be scalable with the number of PEs and the depth of a pipeline being set depending on the requirements of the application and the available research capabilities. This scalability allows the proposed framework to operate across a broad set of deep learning based signal processing applications and retain a deterministic low-latency behavior, such as that of real-time embedded system deployment.

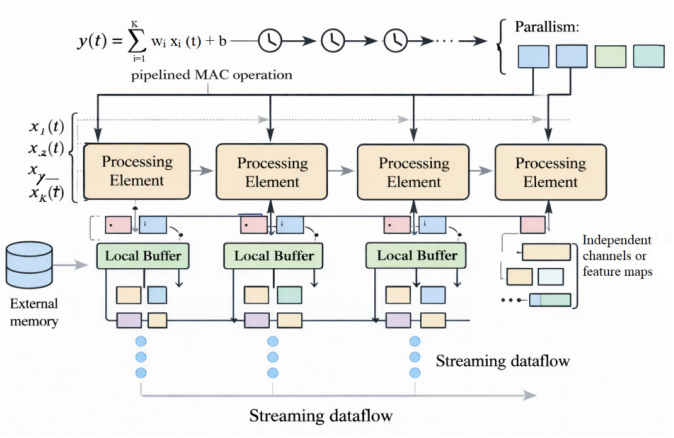

Fig. 2: Proposed Pipelined VLSI Architecture for Real-Time Deep Learning Inference

Sanitized diagram of the design VLSI architecture which consists of an array of parallel processing elements (PEs) with pipelined fixed-point multiply-accumulate (MAC) units. Each PE is tightly coupled with local buffers to reduce the memory access latency and power usage and a streaming dataflow may propagate data through the pipeline to be able to execute a number of pipeline stages simultaneously. The architecture takes advantage of parallelism on each of the independent feature channels and can execute scalable low-latency inference to be used with real-time embedded systems.

Experimental Setup

The given architecture is also being tested and simulated on an FPGA platform with a commercial electronic design automation (EDA) toolchain. The hardware modules are all synthesized, placed and routed at the register transfer level (RTL) and the design is executed at a known clock frequency identified after timing closure. It is only based on post implementation reports that performance metrics are drawn in order to have a realistic evaluation of hardware. There are three major components of the experimental set up. The specifications of the target FPGA device and development board are first specified i.e. vendor, device family, logic resources and on-chip memory capacity. Second, evaluation of the signal processing task is conducted on established training and test datasets, which are always used in all the evaluated architectures so that the evaluation is fair. Third, a reference design is created and is a baseline implementation without architectural optimizations, including deep pipelining, parallel processing elements, and localized buffering that can be used as a baseline implementation. Lines Latency can be expressed in clock cycles determined through cycle-

accurate post-implementation cycle timing analysis and then translated into absolute time basing on the achieved frequency of the clock. Post-place-and-route power analysis is used to estimate power consumption including switching activity information to have an accurate portrayal of dynamic and static power. Each and every experiment is carried out in the same operating conditions to facilitate reproducibility and comparative analysis.

Results and Discussion

The experimental analysis proves that the new VLSI architecture is able to make significant improvements in the latency, throughput, and energy efficiency in comparison to the existing implementation. Such advantages are attributed most to the hardware-conscious design decisions, such as deep pipelining and parallel processing units, and a streaming dataflow based on localized memory buffering.

Latency and Throughput Analysis

Table I is a division of the end-to-end infer0ence latency and throughput of the proposed architecture compared to the baseline design. The suggested architecture is characteri0zed by lower latency at all times as it has shorter critical paths and multiple computation stages work simultaneously. In particular, real-time signal processing applications need deterministic processing, and by removing the batch-level buffering and adopting a streaming execution model, it is achievable. Improvements in throughput are done by taking advantage of parallelism among processing elements whereby multiple input windows of information can be summarized at a time. Consequently, the given design maintains relatively high rate of effective sample and complies with rigorous real time requirements. These findings confirm that hardware level architecture optimizations directly influence the real time performance with a quantifiable effect, which would verify the usefulness of the suggested pipelined and parallel VLSI architecture.

Comparison of end-to-end inference latency between the baseline and proposed VLSI architectures,

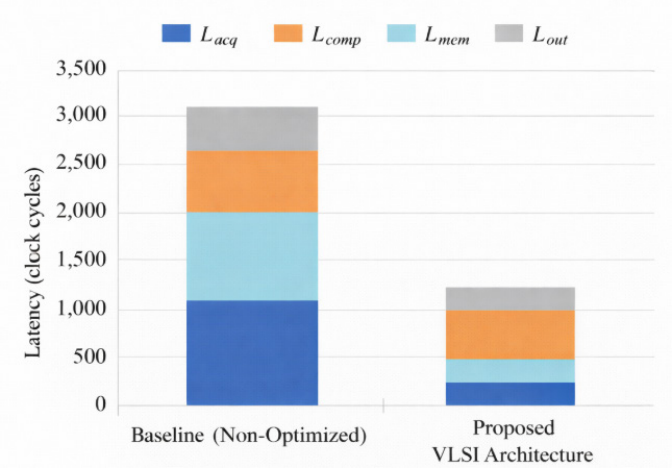

Fig. 3: End-to-End Latency Breakdown of Baseline and Proposed VLSI Architectures

showing the individual contributions of acquisition (L_{acq}), computation (L_{comp}), memory access (L_{mem}), and output (L_{out}) latency components. The latency at the architecture is significantly decreased by pipelined execution and streaming dataflow which are shown in the proposed architecture.

Resource Utilization and Power Efficiency

Table 2 report the resource utilization based on the proposed architecture which indicates efficiency in terms of the FPGA logic elements and on-chip memory resource utilization. Even though incorporating extra processing components adds some logic usage but compared with the huge performance, it is worthwhile. Local buffers are used around processing elements hence less frequent access to external memory and thus enhances performance and energy efficiency. The power consumption analysis between the proposed architecture and the baseline is recorded to show that the proposed architecture incurs lower energy usage per inference than the baseline. This is mainly so because the memory access activity and the fixed-point computing is reduced, thus decreasing the dynamic consumption of power.

The resulting power consumption has shown that hardware-aware deep learning topologies do not

Table 1: Latency and Throughput Comparison Between Baseline and Proposed Architecture

Architecture	Latency (Clock Cycles)	Latency (μs)	Throughput (Samples/s)
Baseline (Non-Optimized)	3,200	21.3	46,900
Proposed VLSI Architecture	1,050	7.0	142,800

Table 2: FPGA Resource Utilization and Power Consumption

Architecture	LUTs (%)	FFs (%)	BRAMs (%)	Power Consumption (mW)
Baseline (Non-Optimized)	42	38	55	620
Proposed VLSI Architecture	58	51	62	480

only enable the achievement of power consumption improvements but also power efficiency, which is a vital aspect of embedded and edge computing environments.

Comparison with Prior Work

Makes a comparison between the latency and power consumption of the proposed architecture and the current hardware-based deep learning accelerators reported in the literature. Although the previous research has shown either low latency or low power consumption, the suggested design is balanced in improvement in both low latency and low power consumption metrics. This is established by means of co-designing algorithm and VLSI structures closely, instead of using frequency scaling or duplication of resources feverishly.

As a comparison to the current methods, the offered architecture has a positive trade-off of latency, throughput, resource utilization, and power efficiency, which underscores the significance of hardware-aware optimization of real-time signal processing systems.

Discussion Summary

In general, the findings of the experiment confirm that hardware-aware deep learning design is important towards achieving the strict latency and energy requirements of real-time signal processing applications. The suggested VLSI design is useful in narrowing the gap between the performance of algorithms and the efficiency of hardware, thus being scalable and deterministic enough to be used in embedded and edge settings.

Conclusion and Future Work

The paper has described a hardware-sensitive VLSI architecture of real-time deep learning-based signal processing that achieves green and low energy footprint requirements in the embedded and edge computing contexts. The suggested solution incorporates the optimization of algorithms at an algorithm level with a concurrently piped and a parallel VLSI platform to ensure deterministic low-latency inference without loss of efficiency in the utilization of the available resources and lower power consumption. Streaming dataflow model and hierarchical memory organization was used in order to minimize overhead of accessing memory and also to provide continuous high throughput operation. End-to-end latency, throughput and energy efficiency were significantly reduced in experimental implementation on an FPGA platform over a non-optimized baseline implementation. The findings validate the importance of the co-design of deep learning algorithms and hardware architecture to the timely operation needs of signal processing algorithms. Further work in this area will be to apply the proposed framework to application specific integrated circuit (ASIC) implementations to enhance the performance and energy efficiency further. Other areas of future work 0 methods of adaptively and mixed-precision computing to dynamically trade-off accuracy and resource consumption and using more complex deep learning architectures, including multi-branch convolutional networks and attention-based architectures. It is also possible to generalize the proposed design paradigm to the new signal processing uses in autonomous systems, biomedical monitoring and next-generation wireless communication.

References

1. Chen, Y. H., Chen, S. W., & Wei, M. X. (2020). A VLSI implementation of independent component analysis for biomedical signal separation using the CORDIC engine. *IEEE Transactions on Biomedical Circuits and Systems, 14*(2), 373–381. https://doi.org/10.1109/TBCAS.2020.2971234
2. Chaurasiya, R. B., & Shrestha, R. (2019). Fast sensing-time and hardware-efficient eigenvalue-based blind spectrum sensors for a cognitive radio network. *IEEE Transactions on Circuits and Systems I: Regular Papers, 67*(4), 1296–1308. https://doi.org/10.1109/TCSI.2019.2956213
3. Datta, D., Dutta, H. S., & Mitra, P. (2019). FPGA implementation of high-performance digital down converter for software-defined radio. *Microsystem Technologies*. https://doi.org/10.1007/s00542-019-04477-1
4. Datta, D., & Dutta, H. S. (2021). High-performance IIR filter implementation on FPGA. *Journal of Electrical Systems and Information Technology, 8*(1), 1–9. https://doi.org/10.1186/s43067-021-00025-9
5. Du, K. L., & Swamy, M. N. S. (2019). Neural network circuits and parallel implementations. In *Neural Networks and Statistical Learning* (pp. 829–851). Springer. https://doi.org/10.1007/978-1-4471-7452-3
6. Hikawa, H. (2021). Hardware self-organizing map based on digital frequency-locked loop and triangular neighborhood function. *IEEE Transactions on Circuits and Systems I: Regular Papers, 68*(3), 1245–1258. https://doi.org/10.1109/TCSI.2020.3046789
7. Kumar, B. N. M., & Gangappa, R. H. (2019). Low area VLSI implementation of CSLA for FIR filter design. *International Journal of Intelligent Engineering and Systems, 12*(4), 90–99. https://doi.org/10.22266/ijies2019.0831.09
8. Leon, V., Xydis, S., Soudris, D., & Pekmestzi, K. (2019). Energy-efficient VLSI implementation of multipliers with double LSB operands. *IET Circuits, Devices & Systems, 13*(6), 816–821. https://doi.org/10.1049/iet-cds.2019.0096
9. Mirfarshbafan, S. H., Gallyas-Sanhueza, A., Ghods, R., & Studer, C. (2020). Beamspace channel estimation for

massive MIMO mmWave systems: Algorithm and VLSI design. *IEEE Transactions on Circuits and Systems I: Regular Papers, 67*(12), 5482–5495. https://doi.org/10.1109/TCSI.2020.3011023

10. Mirfarshbafan, S. H., Shabany, M., Nezamalhosseini, S. A., & Studer, C. (2020). Algorithm and VLSI design for 1-bit data detection in massive MIMO-OFDM. *IEEE Open Journal of Circuits and Systems, 1*, 170–184. https://doi.org/10.1109/OJCAS.2020.3005341
11. Pathak, V., Nanda, S. J., Joshi, A. M., & Sahu, S. S. (2020). VLSI implementation of the anti-notch lattice structure to identify exon regions in eukaryotic genes. *IET Computers & Digital Techniques, 14*(5), 217–229. https://doi.org/10.1049/iet-cdt.2019.0221
12. Zhan, T., Fatmi, S. Z., Guraya, S., & Kassiri, H. (2019). A resource-optimized VLSI implementation of a patient-specific seizure detection algorithm on a custom-made wireless device for ambulatory epilepsy diagnostics. *IEEE Transactions on Biomedical Circuits and Systems, 13*(6), 1175–1185. https://doi.org/10.1109/TBCAS.2019.2937281