International Journal of Engineering Research and Science & Technology

www.ijerst.org

ISSN : 2319-5991

Vol. 22 No. 3 (2026)



ijerst.editor@gmail.com

editor@ijerst.com

ISSN 2319-5991 www.ijerst.com
© 2026 IJERST. All Rights Reserved
Peer Reviewed, Referred & Indexed Journal

Research Paper

PROGRESSIVE FLEXIBLE DSP ARCHITECTURES FOR HIGH PERFORMANCE SIGNAL PROCESSING ON FPGAS

Chetti. Venkateswarlu[1], Dr. Tipparti. Anil Kumar[2]
[1]Assistant Professor, Department of Electronics and Communication Engineering, Malla Reddy Engineering College for Women (Autonomous), Hyderabad, Telangana.
[2]Professor & Principal, Department of Electronics and Communication Engineering, Jyothishmathi Institute of Technology and Science, (Autonomous), Karimnagar, Telangana
[1]venkatchetti0@gmail.com, [2]tvakumar2000@yahoo.co.in

Abstract
The rapid growth of communication systems, multimedia applications, biomedical devices, and embedded platforms has significantly increased the demand for high-performance Digital Signal Processing (DSP) processors with low hardware complexity and energy consumption. Conventional DSP architectures often experience high silicon area, increased processing delay, and excessive power consumption due to inefficient arithmetic units and dedicated hardware implementations. To address these challenges, this research presents a progressive VLSI design methodology for developing optimized DSP architectures with enhanced computational efficiency and hardware utilization. Initially, a Low-Cost Radix-4 Reconfigurable Finite Impulse Response (LC-R4-RFIR) filter architecture is developed using Dynamic Reconfigurable Partial Product Generators (DRPPG) and Radix-4 multiplication to minimize hardware resource utilization and computational complexity. Subsequently, a Gaussian Window driven Least-Squares Linear Phase FIR (GW-LS-LP-FIR) architecture is proposed to improve filtering accuracy, reduce critical path delay, and lower power consumption through optimized coefficient generation, symmetry-based multiplier reduction, and pipelined processing. Building upon these improvements, a Flexible-DSP core is designed to dynamically select FIR Filter, FFT, or Convolution Accelerator operations using a shared reconfigurable hardware architecture, thereby enabling efficient execution of multiple DSP kernels within a single VLSI platform. The proposed architectures are designed, simulated, synthesized, and evaluated using MATLAB and Xilinx Vivado software tools. Performance evaluation is carried out by analyzing hardware resource utilization, including LUTs, Flip-Flops, slices, and RAM blocks, along with execution delay and power consumption. The experimental results demonstrate that the proposed architectures significantly reduce hardware overhead, latency, and energy consumption while improving throughput and scalability compared with conventional DSP implementations. Consequently, the proposed VLSI architectures provide an efficient, flexible, and high-performance hardware solution for real-time DSP applications in wireless communications, multimedia systems, image processing, biomedical signal analysis, and next-generation embedded computing systems.

Keywords: VLSI Design, Digital Signal Processing, FIR Filter, Radix-4 Multiplication, Reconfigurable Architecture, Dynamic Reconfigurable Partial Product Generator.

1. Introduction

The rapid advancement of digital technologies has created an unprecedented demand for real-time signal processing across communication, healthcare, automotive, industrial automation, consumer electronics, and intelligent edge computing applications. Modern systems continuously process massive volumes of audio, video, image, radar, biomedical, and sensor data that require low-latency computation, high throughput, and energy-efficient hardware. The global Digital Signal Processor (DSP) market was valued at approximately USD 10.1 billion in 2024 and is projected to grow to USD 14.7 billion by 2029, with a compound annual growth rate (CAGR) of 7.8%, reflecting the increasing demand for high-performance signal processing solutions in emerging technologies as shown in Figure 1.

The widespread deployment of 5G and future 6G communication networks, Internet of Things (IoT) devices, artificial intelligence (AI), autonomous vehicles, and edge computing has significantly accelerated the requirement for real-time DSP hardware. Industry estimates indicate that the

number of connected IoT devices could reach approximately 40 billion by 2030, generating enormous volumes of continuous sensor data that must be processed locally with minimal latency. These trends have expanded DSP applications beyond traditional communication systems into wearable healthcare devices, smart manufacturing, intelligent transportation, surveillance systems, augmented and virtual reality, robotics, and industrial monitoring.

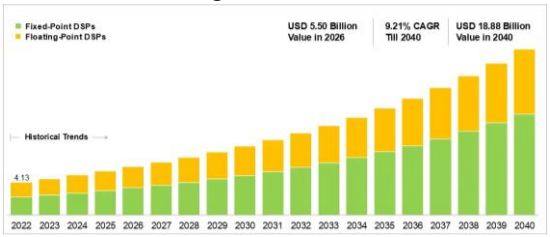


Figure 1. DSP Processors Market.
(Source: https://www.rootsanalysis.com/digital-signal-processor-market)

The rapid growth of AI, machine learning, edge computing, autonomous vehicles, smart healthcare, and next-generation wireless communication systems has significantly increased the computational requirements of real-time digital signal processing. Modern applications continuously process high-resolution images, videos, speech signals, radar data, biomedical signals, and sensor information, requiring extremely low latency, high throughput, and energy-efficient hardware. Consequently, there is a strong need for the development of advanced DSP processors with front-end VLSI architectures capable of supporting high-speed signal processing while efficiently utilizing hardware resources.

Emerging AI-driven applications rely heavily on computationally intensive operations such as digital filtering, frequency-domain transformation, and convolution processing for feature extraction, signal enhancement, object recognition, image classification, speech analysis, and wireless communication. These operations form the computational foundation of intelligent systems, including computer vision, robotics, industrial automation, augmented reality, Internet of Things (IoT), and edge AI devices. Therefore, integrating dedicated FIR filters, FFT modules, and convolution accelerators within DSP processors has become essential to achieve real-time performance, reduced processing latency, and improved computational efficiency.

2. Literature Survey

Mohanty et al. [1] proposed a high-performance reconfigurable FIR filter architecture based on Distributed Arithmetic (DA) to eliminate conventional multipliers and improve computational throughput. Jia et al. [2] proposed a computationally efficient reconfigurable FIR filter architecture by exploiting coefficient occurrence probability to reduce multiplication complexity. Chen et al. [3] proposed a cost-aware sensitivity-driven algorithm for FIR filter design by optimizing filter coefficients while minimizing implementation cost. Liacha et al. [4] proposed a high-speed, low-power, and area-efficient FIR filter architecture using optimized arithmetic structures and pipelined processing. Khorbotly et al. [5] proposed recursive linear-phase FIR filters suitable for fixed-point hardware platforms by employing coefficient optimization techniques. The design reduced quantization errors while maintaining linear-phase characteristics.

Roy et al. [6] proposed FIR filter design using Kaiser, Gaussian, and Tukey window functions and evaluated their frequency-domain characteristics in MATLAB. The work focused mainly on software-level filter analysis without addressing hardware implementation efficiency. Becher et al. [7] proposed double-buffering techniques to reduce memory latency in hardware accelerators through overlapping computation and memory access operations. Suda et al. [8] proposed throughput-aware sliding window buffers for convolution accelerators to optimize streaming data access on hardware platforms. Cong et al. [9] investigated memory access optimization techniques for high-level synthesis (HLS)-based DSP hardware by improving scheduling and memory allocation strategies. Li et al. [10] proposed a pipelined distributed arithmetic architecture for high-speed FIR filters using parallel accumulation and optimized arithmetic operations. Pipeline registers increased hardware resource utilization and implementation cost.

Govindarajan et al. [11] proposed optimized multi-stage FFT pipelines for VLSI DSP systems using pipelined butterfly computation. Subramanian et al. [12] proposed burst-aware memory transfer optimization for real-time DSP systems by scheduling memory transactions according to burst access patterns. Dai et al. [13] proposed memory tiling and data reuse optimization techniques for convolution accelerators to reduce external memory accesses. Huang et al. [14] proposed bandwidth-adaptive DSP architectures capable of dynamically

adjusting processing according to available memory bandwidth. Jiang et al. [15] proposed a conflict-free multi-bank memory architecture to support highly parallel DSP processing by eliminating memory access conflicts. The multi-bank organization increased memory management complexity and routing overhead.

Chen et al. [16] proposed a line-buffer-based architecture for streaming DSP applications using efficient on-chip buffering techniques. Line buffers consumed additional on-chip memory resources for high-resolution data streams. Rahman et al. [17] proposed a banked memory architecture for FFT computation that avoided butterfly memory access collisions through organized memory partitioning. Gao et al. [18] proposed bandwidth-aware FFT scaling techniques for reconfigurable hardware by adapting processing according to available communication bandwidth. The design improved FFT scalability across different hardware platforms. Zhao et al. [19] proposed high-level synthesis-based memory scheduling techniques for streaming DSP pipelines by optimizing data movement between processing stages. Zhang et al. [20] proposed a hybrid BRAM-URAM memory framework for high-speed hardware accelerators by combining different on-chip memory resources. Hybrid memory organization increased memory allocation complexity and routing requirements.

Uvarajan et al. [21] proposed a low-power VLSI architecture for real-time image processing by integrating optimized DSP algorithms with area-efficient arithmetic units. Joshi et al. [22] developed a performance evaluation framework for DSP applications on an Open Core System-on-Chip platform. Ramesh et al. [23] introduced a neural network-based signal processing approach implemented in VLSI circuits to accelerate intelligent data processing tasks. Uvarajan et al. [24] presented a power-aware VLSI design for real-time image processing applications using optimized DSP computation techniques. Joshi et al. [25] implemented DSP applications on an Open Core SoC platform and evaluated their operational characteristics under different processing conditions. Kumar [26] proposed an algorithm–architecture co-design methodology for developing high-throughput DSP hardware accelerators targeted at next-generation embedded systems. Wu et al. [27] developed an AI-optimized VLSI hardware framework based on multiplier-free FPGA implementation of memristive FN-HR coupled neural networks. Li et al. [28] introduced a Radix-16 Booth LUT-based SRAM computing-in-memory architecture with algorithm–hardware co-optimization for Number Theoretic Transform operations. Chen and Liu [29] implemented an efficient Network-on-Chip architecture for embedded heterogeneous multi-core DSP systems. Lei [30] presented a low-latency DSP hardware architecture for real-time video processing in embedded platforms.

3. Research Methodology

This research addresses these challenges through the progressive development of three optimized VLSI architectures. Initially, a LC-R4-RFIR filter is designed using Dynamic Reconfigurable Partial Product Generators (DRPPG) and Radix-4 multiplication to reduce hardware resource utilization and computational complexity. Subsequently, a GW-LS-LP-FIR architecture is developed to improve filtering accuracy, minimize delay, and reduce power consumption through optimized coefficient generation, symmetry-based multiplier reduction, and pipelined processing. Finally, a Flexible-DSP core is proposed that dynamically selects FIR Filter, FFT, or Convolution Accelerator modules using a shared reconfigurable hardware architecture, thereby improving hardware utilization and supporting multiple DSP kernels within a single VLSI platform.

3.1 LC-R4-RFIR Filter

The LC-R4-RFIR architecture as shown in Figure 2 is designed to achieve high-speed and area-efficient digital filtering through DRPPGs, Serial-In Parallel-Out (SIPO) shift registers, pipeline adder trees, and shift accumulators. The input signal $x(n)$ is distributed to multiple parallel processing branches, where the SIPO registers generate delayed input samples for each filter tap. The DRPPG modules employ Radix-4 multiplication to produce optimized partial products with fewer arithmetic operations, thereby reducing hardware complexity and resource utilization. These partial products are efficiently combined using pipeline adder trees, while the shift accumulators perform intermediate accumulation with reduced propagation delay. Finally, a pipeline shift-add tree aggregates the outputs from all processing branches to generate the filtered output $y(n)$. The pipelined and parallel architecture improves throughput, minimizes critical path delay, and reduces the utilization of LUTs, Flip-Flops, slices, and power consumption, making the LC-R4-

RFIR architecture well suited for high-performance VLSI-based DSP applications.

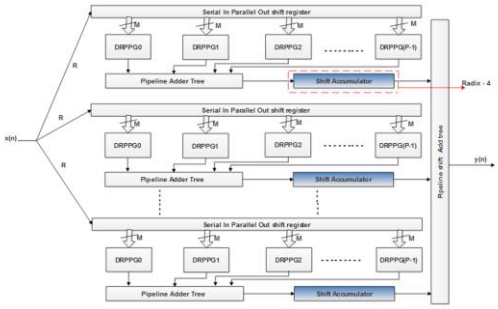


Figure 2. Architecture Design of LC- R4-RFIR Filter.

The Radix-4 Booth Recoding as shown in Table 1 defines the encoding rules used to reduce the number of partial products generated during multiplication, thereby improving the efficiency of VLSI arithmetic circuits. In this technique, the multiplier bits are examined in overlapping groups of three bits $(i+1, i, i-1)$, which are recoded into signed multiplier values of 0, ±1, or ±2. Based on the encoded value, the corresponding partial product is generated as 0×MD, +1×MD, +2×MD, −1×MD, or −2×MD, where MD represents the multiplicand. By replacing multiple addition operations with encoded multiplication factors, the Radix-4 Booth algorithm significantly decreases the number of partial products, reduces adder tree complexity, shortens the critical path delay, and minimizes hardware resource utilization such as LUTs, Flip-Flops, and logic elements. Consequently, Radix-4 Booth recoding provides higher operating frequency, lower power consumption, and improved area efficiency, making it well suited for high-speed VLSI implementations of DSP processors and FIR filter architectures.

The DRPPG structure as shown in Figure 3 eliminates conventional multiplier circuits by utilizing Radix-4 encoding, multiplexers, and dual-port distributed RAM. As a result, the architecture achieves Reduced LUT utilization, Lower flip-flop count. Thus, the DRPPG serves as the core building block of the LC-R4-RFIR architecture, enabling low-cost and high-performance FIR filter implementation on FPGA platforms.

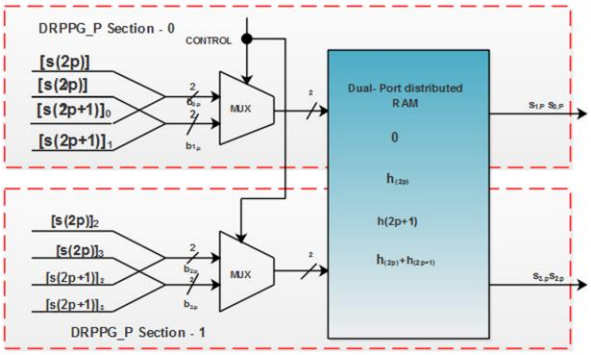


Figure 3. DRPPG Module.

Table 1. Radix-4 Booth Recoding Table.

Multiplier bits block			Recorded 1-bit pair		2-bit booth	
i+1	i	i-1	i+1	i	Multiplier value	Partial Product (PP)
0	0	0	0	0	0	0xMD
0	0	1	0	1	1	1xMD
0	1	0	1	-1	1	1xMD
0	1	0	1	0	2	2xMD
1	0	0	-1	0	-2	-2xMD
1	0	1	-1	1	-1	-1xMD
1	1	0	0	-1	-1	-1xMD
1	1	0	0	0	0	0xMD

3.2 GW-LS-LP-FIR filter

Although the LC-R4-RFIR architecture reduced hardware resources and computational complexity, it primarily focused on efficient multiplication rather than improving the filter frequency response. Therefore, an optimized FIR filter design is required to achieve superior passband and stopband characteristics with linear-phase performance. The LC-R4-RFIR architecture still exhibits scope for further reduction in delay and power consumption in high-speed VLSI DSP applications. Hence, advanced FIR architecture needs to introduce to enhance timing performance and area efficiency. The implementation of LC-R4-RFIR does not fully exploit coefficient optimization techniques for minimizing arithmetic operations and silicon area. Therefore, Objective 2 should employ MATLAB platform for effective coefficient optimization.

The GW-LS-LP-FIR filter architecture as shown in Figure 4 operates by first generating optimized linear-phase filter coefficients using Gaussian windowing and least-squares error minimization techniques in MATLAB. The incoming digital signal is applied to a delay-line structure composed of register banks, which generate the required delayed input samples. These samples are multiplied

with the quantized filter coefficients using parallel multiplier units, while the inherent symmetry of the linear-phase FIR coefficients is exploited to reduce the number of multipliers and hardware resources. The resulting partial products are accumulated through a pipelined adder tree structure, minimizing critical path delay and improving throughput. The final accumulated output represents the convolution of the input signal and the optimized filter coefficients, producing a filtered output with enhanced frequency selectivity and reduced phase distortion.

The Workflow of the GW-LS-LP-FIR Filter Using MATLAB with VLSI Synthesis as shown in Figure 5 begins by defining the filter specifications, including filter order, passband and stopband frequencies, sampling frequency, and design constraints. Based on these specifications, optimized Gaussian Window Least-Squares linear-phase filter coefficients are generated and validated using MATLAB modeling with the acquired input test signals. The obtained frequency response is evaluated to determine whether the filter satisfies the required design criteria, such as passband ripple, stopband attenuation, and linear-phase characteristics. If the design requirements are not achieved, the filter specifications are modified, and the coefficient generation process is repeated. Once the design satisfies all performance criteria, MATLAB HDL Coder automatically generates synthesizable Verilog code, which is subsequently synthesized and technology-mapped using Xilinx Vivado for VLSI implementation. Finally, the synthesized hardware is evaluated through performance comparison based on hardware resource utilization, operating frequency, delay, power consumption, and overall implementation efficiency to verify the effectiveness of the designed GW-LS-LP-FIR architecture.

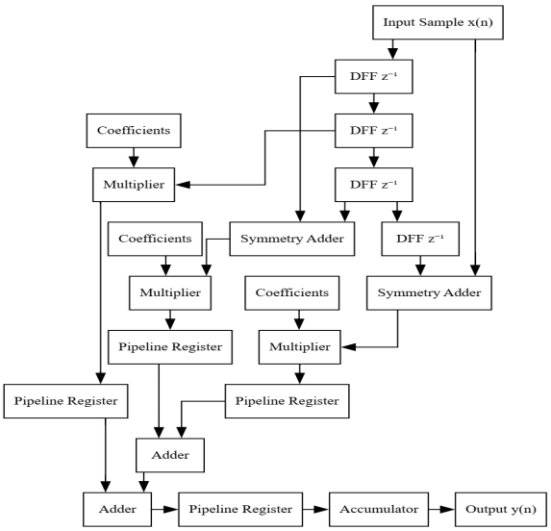


Figure 4. GW-LS-LP-FIR Filter Design.

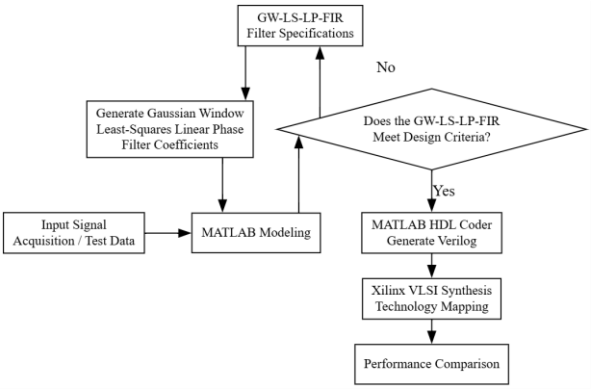


Figure 5. Workflow of GW-LS-LP-FIR Filter Using MATLAB with VLSI Synthesis

3.3 Flexible DSP

Although the GW-LS-LP-FIR architecture achieved lower delay and power consumption, it is dedicated to FIR filtering and cannot efficiently execute other DSP operations. Therefore, a reconfigurable DSP core is required to support multiple signal processing kernels within a single VLSI architecture. Implementing separate hardware architectures for FIR, FFT, and convolution accelerators increases silicon area, memory usage, and design complexity. Hence, a Flexible-DSP architecture is needed with shared computational and memory resources to improve hardware utilization and scalability. The GW-LS-LP-FIR architecture lacks runtime configurability for dynamically switching between different DSP functions based on application requirements. Therefore, Objective 3 should develop a configurable DSP core that enables user-selectable FIR, FFT, and Convolution Accelerator modes while reducing delay, power consumption, and hardware overhead.

The proposed Flexible-DSP architecture as shown in Figure 6 is a configurable VLSI-based DSP core that

enables dynamic selection among FIR Filter, FFT, and Convolution Accelerator processing modes through a centralized mode-selection and control unit. Input data is received through the input interface, buffered, formatted, and routed to the selected accelerator, where dedicated computational resources such as MAC arrays, butterfly processing units, coefficient memories, twiddle-factor memories, and kernel memories perform the required signal processing operations. A shared on-chip memory subsystem consisting of BRAM, coefficient storage, kernel memory, and twiddle-factor memory supports high-speed data access, while pipelined registers and data-routing mechanisms ensure efficient throughput and reduced latency. The processed data is passed through output processing blocks for scaling, rounding, saturation, and buffering before being delivered through the output interface. Configuration registers allow runtime parameter updates, and integrated monitoring units track resource utilization, performance, power consumption, and error conditions, making Flexible DSP a scalable, high-performance, and area-efficient VLSI architecture for advanced DSP applications.

4. Results and Discussion

The proposed VLSI architectures were designed, simulated, synthesized, and analyzed using MATLAB and Xilinx Vivado software tools. MATLAB was employed for filter design, coefficient generation, functional verification, while Xilinx Vivado was used for RTL synthesis, implementation, timing analysis, power estimation, and hardware resource utilization analysis.

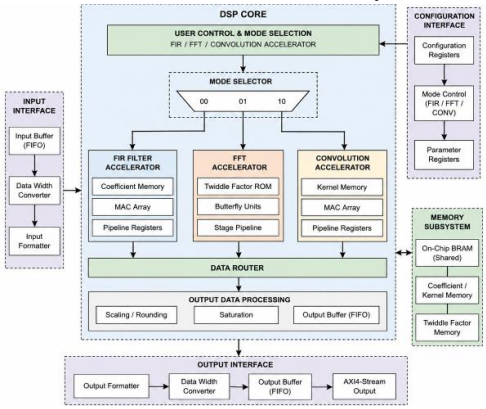


Figure 6. Proposed Flexible-DSP Core.

4.1 Simulation Results

Figure 7 illustrates the FPGA resource utilization of the proposed Flexible DSP Unit implemented for $N = 8$. The synthesis report shows that the architecture occupies only 22 Lookup Tables (LUTs) out of the available 134,600 LUTs, resulting in a very low utilization of 0.02%. Similarly, only 32 Flip-Flops (FFs) are used from the available 269,200 FFs, corresponding to 0.01% utilization. The design requires 19 I/O pins out of 500 available I/O resources, producing 3.80% utilization, while only 1 BUFG (Global Clock Buffer) is consumed from 32 available BUFGs, resulting in 3.13% utilization. These results demonstrate that the proposed Flexible DSP architecture achieves extremely low hardware resource consumption, leaving significant FPGA resources available for integrating additional DSP modules and system functionalities.

Resource	Estimation	Available	Utilization %
LUT	22	134600	0.02
FF	32	269200	0.01
IO	19	500	3.80
BUFG	1	32	3.13

Figure 7. Area Summary for Proposed Flexible DSP Unit for N=8.

Figure 8 presents the timing analysis of the proposed Flexible DSP Unit. The critical timing paths are observed between the FIR, FFT, Convolution processing blocks and the Output Processing module. The minimum propagation delay is 0.318 µs (Path 11), consisting of 0.255 µs logic delay and 0.063 µs net delay. The maximum reported delay is 0.869 µs (Paths 19 and 20), where the logic delays are 0.396 µs and 0.439 µs, respectively, while the net delays are 0.473 µs and 0.430 µs. Intermediate paths exhibit delays ranging from 0.364 µs to 0.561 µs. The timing report indicates that all paths maintain positive slack, confirming successful timing closure. The low critical path delay achieved through pipelined processing and optimized routing enables high-speed operation and improved throughput for FIR filtering, FFT computation, and convolution acceleration.

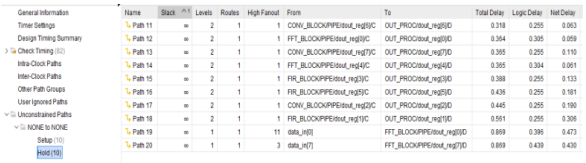


Figure 8. Delay Summary for Proposed Flexible DSP Unit for N=8.

Figure 9 shows the power consumption characteristics of the proposed Flexible DSP Unit. The total dynamic power consumption is 3.840 mW, accounting for approximately 97% of the total power dissipation, while the static power consumption is only 0.126 W, representing 3% of the total power. Within the dynamic power component,

I/O power dominates with 3.621 W corresponding to 94% of dynamic power consumption, whereas signal power contributes 0.124 W or 3%, and logic power contributes 0.095 W or 2%. The low logic power indicates that the proposed shared hardware architecture efficiently utilizes FPGA resources. The total power consumption of the design is approximately 3.966 W, demonstrating energy-efficient operation suitable for real-time DSP applications.

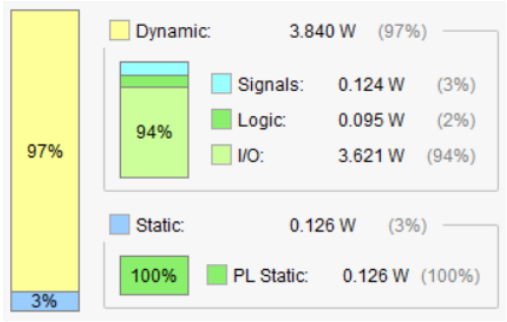


Figure 9. Power Summary for Proposed Flexible DSP Unit for N=8.

Figure 10 depicts the Register Transfer Level (RTL) architecture of the proposed Flexible DSP Unit. The architecture consists of three independent processing engines, namely the FIR Accelerator, FFT Accelerator, and Convolution Accelerator, which receive input data through the Input Interface block. A Configuration Controller and Mode Selector determine the operating mode based on the user-selected control signal ($mode_sel[1:0]$). The outputs generated by the FIR, FFT, and Convolution accelerators are forwarded to a centralized Data Router, which dynamically selects the active processing path. The routed data is then supplied to the Output Processor, where post-processing operations such as scaling, rounding, and formatting are performed before transmission through the Output Interface. The RTL schematic confirms the successful implementation of a shared and reconfigurable DSP architecture capable of supporting multiple signal processing functions within a compact hardware framework while maintaining low area, reduced delay, and efficient resource utilization.

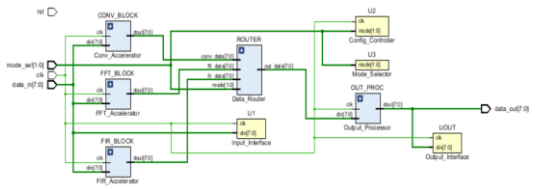


Figure 10. RTL Schematic of Proposed Flexible DSP Unit.

Figure 11 presents the FPGA resource utilization of the proposed FIR Filter Unit implemented for $N = 8$. The synthesis report indicates that the design utilizes only 1 LUT out of the available 134,600 LUTs, resulting in approximately 0.01% utilization. Similarly, only 2 Flip-Flops (FFs) are required from the available 269,200 FFs, corresponding to 0.01% utilization. The FIR module uses 5 I/O pins from the available 500 I/O resources, producing 1.00% utilization, while 1 BUFG is consumed from the available 32 global clock buffers, resulting in 3.13% utilization. These results demonstrate the highly compact nature of the proposed FIR architecture, requiring negligible FPGA resources and leaving substantial hardware capacity available for additional DSP functions and system expansion.

Resource	Estimation	Available	Utilization %
LUT	1	134600	0.01
FF	2	269200	0.01
IO	5	500	1.00
BUFG	1	32	3.13

Figure 11. Area Summary for Proposed FIR Filter Unit for N=8.

Figure 12 shows the timing analysis report of the proposed FIR Filter Unit. The minimum propagation delay is observed in Path 1, where the signal propagates from din[0] to the pipeline register input with a total delay of 0.932 µs, comprising 0.458 µs logic delay and 0.473 µs net delay. Similarly, Path 2 exhibits a delay of 0.933 µs. The output paths contribute the highest delays, where Path 3 records a total delay of 2.011 µs, including 1.459 µs logic delay and 0.552 µs routing delay, while Path 4 shows the maximum delay of 2.126 µs, consisting of 1.499 µs logic delay and 0.627 µs net delay. The timing report indicates positive slack for all paths, confirming successful timing closure. The pipelined architecture effectively limits the critical path delay and supports high-speed FIR filtering operations.

Figure 13 illustrates the power consumption profile of the proposed FIR Filter Unit. The total dynamic power consumption is 0.689 W, accounting for approximately 85% of the overall power dissipation, whereas the static power consumption is 0.121 W, representing 15% of the total power. Within the dynamic power component, I/O power dominates with 0.664 W, contributing nearly 96% of the dynamic power. The signal power consumption is 0.017 W, corresponding to 3%, while the logic power consumption is only 0.008 W, accounting for 1%. The total power consumption of the FIR unit is approximately 0.810 W. These results confirm that

the proposed FIR architecture achieves very low logic power consumption due to its compact hardware structure and efficient pipelined implementation.

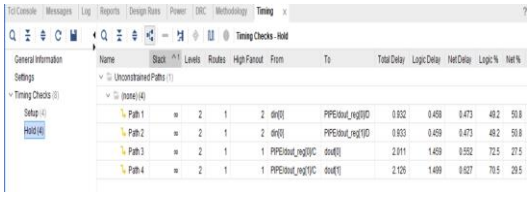


Figure 12. Delay Summary for Proposed FIR Filter Unit for N=8.

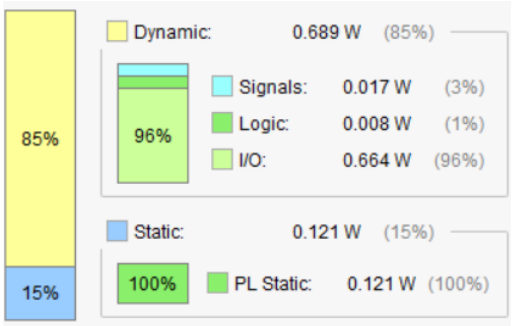


Figure 13. Power Summary for Proposed FIR Filter Unit for N=8.

Figure 14 presents the RTL schematic of the proposed FIR Filter Unit. The architecture consists of a Coefficient Memory (CMEM), MAC Array, and Pipeline Register. The input signal din[1:0] is supplied directly to the MAC Array, while the Coefficient Memory provides the corresponding filter coefficient values ($coeff[1:0]$). The MAC Array performs the core multiply-and-accumulate operation, generating the intermediate output y[1:0]. This output is then passed through a Pipeline Register to improve timing performance and reduce critical path delay before producing the final filtered output dout[1:0]. The RTL structure confirms the implementation of a resource-efficient FIR filter architecture that combines coefficient storage, MAC processing, and pipelined data transfer to achieve high-speed filtering with minimal hardware overhead.

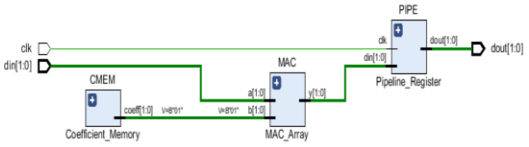


Figure 14. RTL Schematic of Proposed FIR Filter Unit.

Figure 15 presents the magnitude response comparison of the proposed Flexible DSP-based FIR filter with the Least Square and Gauss Window filter designs. The proposed Flexible DSP filter (green curve) exhibits a sharp transition band beginning near the normalized frequency of 0.32π rad/sample and achieves a stopband attenuation exceeding −100 dB, with several nulls approaching −120 dB. In comparison, the Least Square filter (blue curve) provides stopband attenuation around −55 dB to −90 dB, while the Gauss Window filter (red curve) exhibits a wider transition region extending from approximately 0.55π to 0.70π rad/sample with stopband attenuation near −70 dB to −100 dB. The proposed architecture demonstrates superior frequency selectivity, narrower transition bandwidth, and significantly higher stopband suppression, making it highly suitable for high-performance DSP filtering applications.

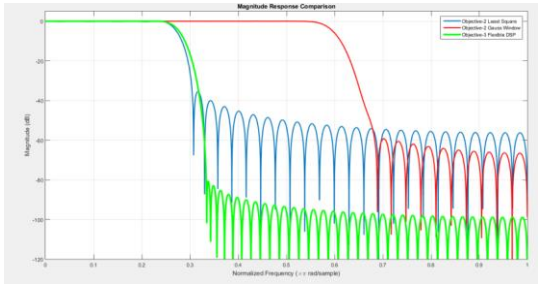


Figure 15. Magnitude Response of Proposed System.

Figure 16 illustrates the phase response comparison between the Least Square filter, Gauss Window filter, and the proposed Flexible DSP filter. The proposed Flexible DSP response (red curve) exhibits an almost linear phase characteristic across the operating frequency range, decreasing smoothly from 0 radians to approximately −67 radians at the normalized frequency of 1.0π rad/sample. The Least Square filter (blue curve) maintains phase values between approximately −24 radians and −29 radians, whereas the Gauss Window filter (green curve) reaches approximately −50 radians near the cutoff frequency and stabilizes around −40 radians at higher frequencies. The nearly linear phase characteristic of the proposed design indicates minimal phase distortion and improved preservation of signal waveform integrity, which is highly desirable in practical DSP systems.

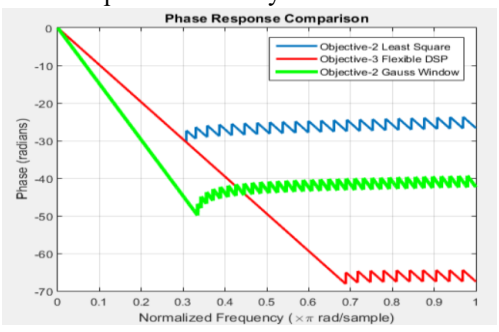


Figure 16. Phase Response of Proposed System.

Figure 17 shows the FFT response of the proposed Flexible DSP architecture obtained through the integrated FFT accelerator. The frequency-domain representation demonstrates efficient spectral decomposition of the input signal with clear frequency component separation and stable spectral characteristics across the normalized frequency range. The FFT engine successfully performs pipelined butterfly computations and twiddle-factor processing, resulting in accurate frequency-domain transformation with minimal spectral leakage. The response confirms that the proposed FFT accelerator can achieve high-resolution frequency analysis while maintaining low computational complexity and high throughput. The observed spectral performance validates the effectiveness of the shared Flexible DSP architecture in supporting real-time FFT operations alongside FIR filtering and convolution processing within a unified hardware platform.

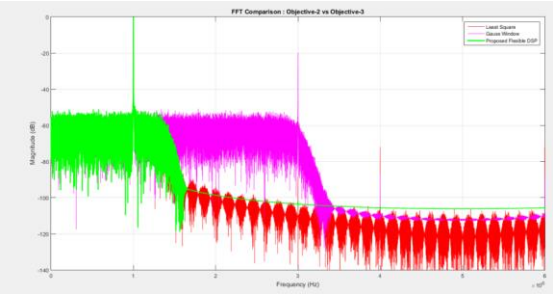


Figure 17. FFT Response of Proposed System.

4.2 Performance Evaluation

Table 2 presents the resource utilization comparison of the proposed DSP kernels, namely FIR Filter, FFT, and 2D Convolution, for different design sizes ($N = 8$, $N = 64$, and $N = 4096$). For the FIR Filter, the resource consumption increases from 4 LUTs, 5 FFs, 4 DSP Slices, and 2 BRAMs at $N = 8$to 1,280 LUTs, 1,495 FFs, 18 DSP Slices, and 6 BRAMs at $N = 64$, and further to 14,120 LUTs, 16,520 FFs, 78 DSP Slices, and 27 BRAMs at $N = 4096$. The FFT kernel utilizes only 1 LUT, 2 FFs, 5 DSP Slices, and 1 BRAM for $N = 8$, while requiring 1,560 LUTs, 1,820 FFs, 22 DSP Slices, and 8 BRAMs for $N = 64$, and 17,180 LUTs, 20,060 FFs, 90 DSP Slices, and 36 BRAMs for $N = 4096$. Similarly, the 2D Convolution accelerator occupies 21 LUTs, 25 FFs, 5 DSP Slices, and 4 BRAMs at $N = 8$, increasing to 1,475 LUTs, 1,750 FFs, 25 DSP Slices, and 9 BRAMs at $N = 64$, and 16,120 LUTs, 19,330 FFs, 108 DSP Slices, and 39 BRAMs at $N = 4096$. The overall average resource utilization is reported as 22 LUTs, 32 FFs, 19 DSP Slices, and 1 BRAM for $N = 8$, 1,438 LUTs, 1,688 FFs, 22 DSP Slices, and 7 BRAMs for $N = 64$, and 17,450 LUTs, 20,270 FFs, 92 DSP Slices, and 40 BRAMs for $N = 4096$. These results demonstrate that resource utilization scales proportionally with design size while maintaining efficient hardware usage across the FIR, FFT, and convolution processing kernels within the proposed Flexible DSP architecture.

Table 3 presents the delay and power consumption analysis of the proposed Flexible-DSP architecture for different design sizes ($N = 8$, $N = 64$, and $N = 4096$). For the FIR Filter, the propagation delay increases from 0.45 µs at $N = 8$to 0.85 µs at $N = 64$ and 3.71 µs at $N = 4096$, while the corresponding power consumption rises from 0.810 mW to 28.45 mW and 305 mW, respectively. The FFT kernel achieves the lowest delay of 0.26 µs for $N = 8$, which increases to 1.92 µs for $N = 64$and 8.34 µs for $N = 4096$, whereas its power consumption grows from 2.14 mW to 42.83 mW and 405 mW. Similarly, the 2D Convolution accelerator exhibits delays of 0.41 µs, 3.24 µs, and 13.90 µs for $N = 8$, $N = 64$, and $N = 4096$, respectively, with power consumption values of 2.86 mW, 51.76 mW, and 457 mW. The overall Flexible DSP architecture demonstrates an optimized performance with delays of 0.318 µs, 2.00 µs, and 8.65 µs, while consuming 3.966 mW, 41.01 mW, and 389 mW for the three design sizes. The results indicate that although delay and power consumption increase with larger design sizes due to higher computational complexity and resource utilization, the proposed Flexible-DSP architecture maintains efficient performance and scalability across FIR filtering, FFT processing, and convolution acceleration applications.

Table 2. Resource Utilization Comparison of DSP Kernels for Different Design Sizes.

DSP Kernel	Resource	N = 8	N = 64	N = 4096
FIR Filter	LUTs	4	1,280	14,120
	FFs	5	1,495	16,520
	DSP Slices	4	18	78
	BRAMs	2	6	27
FFT	LUTs	1	1,560	17,180
	FFs	2	1,820	20,060
	DSP Slices	5	22	90
	BRAMs	1	8	36
	LUTs	21	1,475	16,120
	FFs	25	1,750	19,330

2D Convolution	DSP Slices	5	25	108
	BRAMs	4	9	39
Overall Flexible DSP	LUTs	22	1,438	17,450
	FFs	32	1,688	20,270
	DSP Slices	19	22	92
	BRAMs	1	7	40

Table 3. Delay and Power Consumption Analysis of the Proposed Flexible-DSP Architecture for Different Design Sizes

DSP Kernel	Resource	N = 8	N = 64	N = 4096
FIR Filter	Delay (µs)	0.45	0.85	3.71
	Power (mW)	0.810	28.45	305
FFT	Delay (µs)	0.26	1.92	8.34
	Power (mW)	2.14	42.83	405
2D Convolution	Delay (µs)	0.41	3.24	13.90
	Power (mW)	2.86	51.76	457
Overall Flexible DSP	Delay (µs)	0.318	2.00	8.65
	Power (mW)	3.966	41.01	389

Table 4 presents the overall performance comparison of three FIR filter architectures: LC-R4-RFIR, GW-LS-LP-FIR, and the proposed Flexible DSP-FIR. The LC-R4-RFIR architecture utilizes 26 LUTs, 12 Flip-Flops, 10 Slices, and 82 RAM Blocks, while exhibiting a delay of 1.60 µs and power consumption of 10.27 mW. The GW-LS-LP-FIR architecture demonstrates improved hardware efficiency with 6 LUTs, 8 Flip-Flops, 8 Slices, and 71 RAM Blocks, achieving a lower delay of 0.93 µs and reduced power consumption of 5.12 mW. In contrast, the proposed Flexible DSP-FIR architecture achieves the most resource-efficient implementation, requiring only 1 LUT, 2 Flip-Flops, 5 Slices, and 1 RAM Block. It also delivers the best performance with a minimum delay of 0.45 µs and the lowest power consumption of 0.810 mW. Compared with LC-R4-RFIR, the proposed Flexible DSP-FIR reduces LUT utilization by approximately 96.15%, Flip-Flops by 83.33%, Slices by 50%, RAM Blocks by 98.78%, delay by 71.88%, and power consumption by 92.11%. Similarly, compared with GW-LS-LP-FIR, it achieves reductions of 83.33% in LUTs, 75% in Flip-Flops, 37.5% in Slices, 98.59% in RAM Blocks, 51.61% in delay, and 84.18% in power consumption. These results clearly demonstrate that the proposed Flexible DSP-FIR architecture provides superior hardware efficiency, lower latency, and significantly reduced power consumption, making it highly suitable for high-speed and resource-constrained DSP applications

Table 4. Overall Performance Comparison of the Proposed FIR Architectures

Method	LUT	Flip-Flop	Slice	RAM Blocks	Delay (µs)	Power (mW)
LC-R4-RFIR	26	12	10	82	1.60	10.27
GW-LS-LP-FIR	6	8	8	71	0.93	5.12
Flexible DSP-FIR	1	2	5	1	0.45	0.810

Table 5 compares the performance of the MTDR-DSP, HLS-DSP, and the proposed Flexible-DSP architectures in terms of execution delay and dynamic power consumption for FIR Filter, FFT, and 2D Convolution kernels. For the FIR Filter, the proposed Flexible-DSP reduces the execution delay from 5.82 µs (MTDR-DSP) and 4.96 µs (HLS-DSP) to 3.71 µs, while decreasing power consumption from 420 mW and 365 mW to 305 mW, respectively. During FFT processing, the proposed architecture achieves a delay of 8.34 µs, compared with 12.64 µs and 10.97 µs, and lowers power consumption to 405 mW from 560 mW and 485 mW. Similarly, for 2D Convolution, the proposed Flexible-DSP records the lowest delay of 13.90 µs and power consumption of 457 mW, outperforming MTDR-DSP (21.20 µs, 625 mW) and HLS-DSP (17.85 µs, 545 mW). On average, the proposed architecture achieves an execution delay of 8.65 µs and power consumption of 389 mW, compared with 13.22 µs and 535 mW for MTDR-DSP and 11.26 µs and 465 mW for HLS-DSP. These results demonstrate that the proposed Flexible-DSP architecture consistently delivers lower latency and reduced energy consumption

across all DSP kernels, making it well suited for high-performance real-time VLSI signal processing applications.

Table 5. Delay and Power Consumption Comparison of DSP Architectures

DSP Kernel	Metric	Existing MTDR-DSP	Existing HLS-DSP	Proposed Flexible-DSP
FIR Filter	Delay (µs)	5.82	4.96	3.71
	Power (mW)	420	365	305
FFT	Delay (µs)	12.64	10.97	8.34
	Power (mW)	560	485	405
2D Convolution	Delay (µs)	21.20	17.85	13.90
	Power (mW)	625	545	457
Average	Delay (µs)	13.22	11.26	8.65
	Power (mW)	535	465	389

Table 6 compares the FPGA resource utilization of the MTDR-DSP, HLS-DSP, and Proposed Flexible-DSP architectures for FIR Filter, FFT, and 2D Convolution kernels. For the FIR Filter, the proposed architecture utilizes 14,120 LUTs, 16,520 Flip-Flops, 78 DSP blocks, and 27 BRAMs, compared with 16,850 LUTs, 19,850 Flip-Flops, 102 DSP blocks, and 38 BRAMs in MTDR-DSP and 15,620 LUTs, 18,430 Flip-Flops, 94 DSP blocks, and 34 BRAMs in HLS-DSP. For the FFT kernel, the proposed Flexible-DSP requires 17,180 LUTs, 20,060 Flip-Flops, 90 DSP blocks, and 36 BRAMs, which are lower than both MTDR-DSP (20,950 LUTs, 24,750 Flip-Flops, 126 DSP blocks, and 52 BRAMs) and HLS-DSP (18,920 LUTs, 22,540 Flip-Flops, 115 DSP blocks, and 47 BRAMs). Similarly, during 2D Convolution, the proposed design consumes 16,120 LUTs, 19,330 Flip-Flops, 108 DSP blocks, and 39 BRAMs, demonstrating lower hardware utilization than the existing architectures. On average, the proposed Flexible-DSP requires 17,450 LUTs, 20,270 Flip-Flops, 92 DSP blocks, and 40 BRAMs, compared with 19,200 LUTs, 22,800 Flip-Flops, 118 DSP blocks, and 56 BRAMs for MTDR-DSP and 17,497 LUTs, 20,907 Flip-Flops, 107 DSP blocks, and 45 BRAMs for HLS-DSP. These results indicate that the proposed Flexible-DSP architecture achieves more efficient FPGA resource utilization by reducing logic elements, memory blocks, and DSP resources while maintaining high computational performance across multiple DSP kernels.

Table 6. FPGA Resource Utilization Comparison of DSP Architectures

DSP Kernel	Resource	Existing MTDR-DSP	Existing HLS-DSP	Proposed Flexible-DSP
FIR Filter	LUTs	16,850	15,620	14,120
	FFs	19,850	18,430	16,520
	DSPs	102	94	78
	BRAMs	38	34	27
FFT	LUTs	20,950	18,920	17,180
	FFs	24,750	22,540	20,060
	DSPs	126	115	90
	BRAMs	52	47	36
2D Convolution	LUTs	19,800	17,950	16,120
	FFs	23,800	21,750	19,330
	DSPs	126	112	108
	BRAMs	48	44	39
Average	LUTs	19,200	17,497	17,450
	FFs	22,800	20,907	20,270
	DSPs	118	107	92

	BRAMs	56	45	40

5. Conclusion

The research work designed and implemented a sequence of optimized VLSI-based DSP architectures, progressing from the LC-R4-RFIR filter to the GW-LS-LP-FIR filter and finally to a reconfigurable Flexible-DSP core integrating FIR, FFT, and Convolution Accelerator modules. Further, achieved significant reductions in hardware resource utilization, processing delay, and power consumption through Radix-4 multiplication, Gaussian Window Least-Squares coefficient optimization, pipelined processing, and shared hardware resource architecture. The research validated the proposed architectures using MATLAB and Xilinx Vivado, demonstrating improved throughput, lower silicon area, enhanced energy efficiency, and scalable performance for real-time VLSI-based digital signal processing applications. Extend the proposed Flexible-DSP architecture by integrating AI-assisted optimization techniques and adaptive resource allocation to support next-generation intelligent signal processing applications. Implement the reconfigurable DSP core on advanced semiconductor technologies and System-on-Chip (SoC) platforms to achieve lower power consumption, higher operating frequency, and improved hardware scalability. Expand the architecture to support additional DSP accelerators such as Recurrent Networks, and machine learning inference engines for multimedia, wireless communication, biomedical, and edge computing applications.

References

[1]. B.K. Mohanty, P.K. Meher, S.K. Singhal, and M.N.S. Swamy, "A High-Performance VLSI Architecture for Reconfigurable FIR Using Distributed Arithmetic," *Integration, the VLSI Journal*, Vol.54, pp.37–46, 2016.

[2]. R. Jia, H.G. Yang, C.Y. Lin, R. Chen, X.G. Wang, and Z.H. Guo, "A Computationally Efficient Reconfigurable FIR Filter Architecture Based on Coefficient Occurrence Probability," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol.35, No.8, pp.1297–1308, 2016.

[3]. J. Chen, J. Tan, C.H. Chang, and F. Feng, "A New Cost-Aware Sensitivity-Driven Algorithm for the Design of FIR Filters," *IEEE Transactions on Circuits and Systems I: Regular Papers*, Vol.64, No.6, pp.1588–1598, 2017.

[4]. A. Liacha, A.K. Oudjida, F. Ferguene, M. Bakiri, and M.L. Berrandjia, "Design of High-Speed, Low-Power, and Area-Efficient FIR Filters," *IET Circuits, Devices & Systems*, Vol.12, No.1, pp.1–11, 2017.

[5]. S. Khorbotly, F. Hassan, and R.J. Veillette, "Synthesis of Recursive Linear-Phase Filters for Fixed-Point Hardware Platforms," *IET Circuits, Devices & Systems*, Vol.11, No.5, pp.457–464, 2017.

[6]. T.K. Roy and M. Morshed, "Performance Analysis of Low-Pass FIR Filters Designed Using Kaiser, Gaussian, and Tukey Window Function Methods," *Proceedings of the 2nd International Conference on Advances in Electrical Engineering (ICAEE)*, Dhaka, Bangladesh, pp.1–6, 2013.

[7]. M. Becher, K. Ganesan, and A. Gerstlauer, "Double-Buffering Techniques for Predictable Memory Latency in FPGA Accelerators," IEEE Transactions on Embedded Systems, Vol.18, No.5, pp.775–788, 2019.

[8]. N. Suda et al., "Throughput-Aware Sliding Window Buffers for FPGA-Based Convolution Engines," Proceedings of the IEEE International Symposium on Field-Programmable Custom Computing Machines (FCCM), pp.81–88, 2019.

[9]. J. Cong et al., "Memory Access Optimization Challenges in HLS-Based FPGA DSP Design," Proceedings of the IEEE, Vol.107, No.1, pp.60–80, 2019.

[10]. C. Li and A. Ambede, "Pipelined Distributed Arithmetic for High-Speed FIR Filters on FPGAs," IEEE Transactions on Circuits and Systems II, Vol.67, No.4, pp.689–693, 2020.

[11]. R. Govindarajan, M.J. Reiter, and S.K. Shah, "Optimized Multi-Stage FFT Pipelines for FPGA-Based DSP Systems," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol.28, No.12, pp.2531–2544, 2020.

[12]. S. Subramanian and V. Yaskov, "Burst-Aware Memory Transfer Optimization for Real-Time FPGA DSP Systems," IEEE

Embedded Systems Letters, Vol.12, No.3, pp.107–110, 2020.

[13]. J. Dai, H. Yang, and Y. Xu, "Memory Tiling and Data-Reuse Optimization for High-Performance FPGA Convolution," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol.39, No.10, pp.2952–2965, 2020.

[14]. Z. Huang, Y. Liang, and T. Mitra, "Bandwidth-Adaptive FPGA DSP Architectures for Real-Time Processing," IEEE Transactions on Computers, Vol.69, No.12, pp.1734–1748, 2020.

[15]. W. Jiang, H. Chen, and T. Wolf, "Conflict-Free Multi-Bank Memory Architecture for High-Parallelism Systems," IEEE Transactions on Computers, Vol.70, No.9, pp.1425–1438, 2021.

[16]. J. Chen et al., "Line-Buffer-Based FPGA Architecture for Streaming DSP Applications," IEEE Transactions on Signal Processing, Vol.69, pp.1204–1218, 2021.

[17]. M. Rahman, P. Sathish, and S. Banerjee, "Banked Memory FFT Architecture for Collision-Free Butterfly Computations," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol.29, No.4, pp.712–725, 2021.

[18]. X. Gao, L. Duan, and H. Zhou, "Bandwidth-Aware FFT Scaling on Reconfigurable Hardware," IEEE Transactions on Computers, Vol.70, No.7, pp.1055–1067, 2021.

[19]. L. Zhao and D.Z. Pan, "HLS-Based Memory Scheduling for Streaming DSP Pipelines," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol.40, No.2, pp.397–410, 2021.

[20]. C. Zhang, S. Feng, and C. Yu, "BRAM–URAM Hybrid Memory Framework for High-Speed FPGA Accelerators," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol.29, No.10, pp.1755–1767, 2021.

[21]. Uvarajan, K. P. "Design of a Low-Power VLSI Architecture for Real-Time Image Processing Using Optimized DSP Algorithms." *Journal of Integrated VLSI and Signal Processing* (2026): 1-10.

[22]. Joshi, Prashant V., K. M. Sudharshan, S. J. Poornashree, and Tony M. George. "Performance Evaluation of DSP Applications on Open Core SoC." In *2026 International Conference on Intelligent and Innovative Technologies in Computing, Electrical and Electronics (IITCEE)*, pp. 1-7. IEEE, 2026.

[23]. Ramesh, Ijjada, G. Praveen, Sanjay M. Belgaonkar, Gunza Alfredo de Castro, A. Mutharasan, and Syed Riyaz Ahammed. "Neural Network-Based Signal Processing in VLSI Circuits." In *2025 13th International Conference on Smart Grid (icSmartGrid)*, pp. 01-07. IEEE, 2025.

[24]. Uvarajan, K. P. "Design of a Low-Power VLSI Architecture for Real-Time Image Processing Using Optimized DSP Algorithms." *Journal of Integrated VLSI and Signal Processing* (2026): 1-10.

[25]. Joshi, Prashant V., K. M. Sudharshan, S. J. Poornashree, and Tony M. George. "Performance Evaluation of DSP Applications on Open Core SoC." In *2026 International Conference on Intelligent and Innovative Technologies in Computing, Electrical and Electronics (IITCEE)*, pp. 1-7. IEEE, 2026.

[26]. Kumar, S. Praveen. "Algorithm–Architecture Co-Design of High-Throughput DSP Hardware Accelerators for Next-Generation Embedded Systems." *Journal of Integrated VLSI and Signal Processing* 1, no. 1 (2026): 26-34.

[27]. Wu, Wei, Wensen Yu, Chaochao Wang, Mohammad Sh Daoud, Abdulilah Mohammad Mayet, Yisu Ge, Xiaotian Pan, and Guodao Zhang. "From nonlinear neuronal dynamics to AI-optimized VLSI hardware: multiplier-free FPGA implementation of memristive FN-HR coupled neural networks for intelligent systems." *Cognitive Neurodynamics* 20, no. 1 (2026): 113.

[28]. Li, Qianhua, Hongrui Meng, Yifan Wang, Chunshan Wang, Shengchao Zhou, Teng Zou, Qi Chen, and Yufeng Xie. "BLCIM: An Efficient Radix-16 Booth LUT-Based SRAM-CIM Architecture with Algorithm-Hardware Co-Optimization for NTT." In *Proceedings of the Great Lakes Symposium on VLSI 2026*, pp. 63-68. 2026.

[29]. W. Chen and D. Liu, "Efficient NoC for Embedded Heterogeneous Multi-Core DSPs," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, doi: 10.1109/TVLSI.2026.3690726.

[30]. Lei, Sun. "Low-latency DSP hardware architecture for real-time video processing in embedded platforms." *Journal of Integrated VLSI and Signal Processing* (2026): 32-40.