DESIGN AND DEVELOPMENT OF A HIGH-SPEED, LOW-POWER PIPELINED FIR FILTER USING AN OPTIMISED COMPRESSOR ARCHITECTURE AND ADAPTIVE CLOCK-ENABLE CONTROL

RAJESH DURGAM
Assistant Professor in ECE
rajeshdurgam2001@gmail.com
Malla Reddy College of Engineering

Dr.P.SUDHAKARA REDDY
Associate Professor in ECE
drpsr2022@gmail.com
Srikalahasteeswara institute of technology, Srikalahasti

ABSTRACT
Finite impulse response (FIR) filters are essential parts of digital signal processing systems used in radar, biomedical instruments, wireless communication, audio processing, and image enhancement. Due to the repeated multiplication and accumulation involved in their hardware realisation, size, power, and timing are crucial design limitations. By breaking up a lengthy combinational path into shorter stages, pipelining can increase throughput; however, power and sequential area can be increased by extra registers and clock-related switching. The 16-tap pipelined FIR filter proposed in this study combines an optimised compressor-based partial-product reduction structure with synchronous adaptive clock-enable control. While the compressor network compresses intermediate arithmetic operands prior to final carry propagation, the controller only changes certain pipeline registers when acceptable input data is accepted. The design is designed to be evaluated with Xilinx Vivado 2026 and is described in Verilog HDL.1. The baseline setup is an 8-bit signed input and 8-bit signed coefficient. The motivation and context are established by a discussion of published studies. No numerical improvement is claimed for the suggested architecture since measured synthesis and implementation data for the suggested RTL have not been provided. Resource usage, power, timing, and frequency obtained under the same baseline and suggested design conditions must be reported in the final evaluation.

KEYWORDS
FIR filter, pipelining, synchronous clock enablement, low-power VLSI, Wallace tree, compressor, Verilog HDL, FPGA, and modified Booth encoding.

I. INTRODUCTION

In communication, multimedia, instrumentation, and control systems, digital signal processing (DSP) is frequently used to process sampled data. A digital filter attenuates undesirable components while retaining useful information, changing the frequency content of a discrete-time signal. Because they are naturally stable for finite coefficients and can produce a perfect linear-phase response when their coefficients meet the necessary symmetry criterion, finite impulse response filters are especially significant. [4], [11], [1] The input sequence and a set of coefficients are

subjected to a finite convolution using a K-tap FIR filter. $y[n] = \Sigma(k=0 \text{ to } K-1)\ h[k]x[n-k]$ is its output, where x[n] is the input sample and h[k] is the kth coefficient, and the filtered result is y[n]. K multiplications and K−1 additions are needed for every output in a direct- form of execution. Both the arithmetic hardware and the datapath delay increase with the number of taps and operand widths. (1), [18] Although the direct-form architecture is conceptually straightforward, a lengthy series of additions could lead to a critical path. Adder trees and parallel multiplier architectures minimise serial computing. By inserting registers at predetermined points, pipelining further splits the process into stages. Even though the output shows after multiple cycles of latency, a new input sample can be accepted every clock cycle with the right pipeline scheduling. [4], [5], [1] Timing performance and sequential overhead are the primary trade-offs of a pipelined FIR filter. By lowering the quantity of combinational logic between clock edges, register insertion may enable a higher clock frequency. Simultaneously, every additional register uses FPGA resources and may increase dynamic power through data and clock transitions. Therefore, the activity of its pipeline registers should be taken into account in a design that increases speed. [7], [8], [1] A hierarchical Wallace reduction tree, fine-grained seamless pipelining, and improved Booth encoding are used in Cho et al.'s [1] very-high-speed pipeline FIR filter. In comparison to systolic designs, their article

reports a 73.5% reduced area-delay product and up to 1.8 GSPS. These figures are part of the design and assessment environment of the base article. The suggested extension adds activity-aware register control while maintaining the high-speed arithmetic motive. Data does not always come on every clock cycle in streaming apps. Pipeline data registers might not need to collect new values during idle cycles. Avoidable data-path switching can be decreased if the register bank is only activated when a legitimate transaction advances. The idle-cycle ratio and how the FPGA implements the clock-enable function determine the benefit. [8], [7] Implementing a 16-tap FIR filter with signed 8-bit samples and coefficients, introducing synchronous enable control, organising partial-product accumulation with a compressor-based reduction structure, maintaining output timing by delaying valid information with data, and comparing baseline and suggested architectures using identical Vivado settings are the goals. [7], [16], [1]

II. RELATED WORK

Reconfigurable FIR architectures were introduced by R. Mahesh and A. P. Vinod [2] with the goal of lowering implementation complexity. In order to minimise the arithmetic operations needed by FIR structures, their study employed coefficient-related optimisation. These methods are helpful when the filter can be expressed using common subexpressions or when the coefficients are fixed. In contrast, the current study focuses on register activity and compressor-based accumulation while maintaining a traditional signed fixed-point FIR datapath.

A common-subexpression-elimination technique was presented by R. Mahesh and A. P. Vinod [3] to realise low-complexity higher-order digital filters. By sharing mathematical terms and identifying recurring coefficient patterns, the technique minimises superfluous hardware. The resulting adder graph and time must be assessed for the target device, even if this can lower the multiplier count.

Fully pipelined systolic architectures for FIR computation utilising distributed arithmetic were proposed by P. K. Meher, S. Chandrasekaran, and A. Amira [4]. By choosing the lookup-table address length, their architecture provides a trade-off between area, delay, and power. Although they may increase adder complexity and latency, smaller tables can lower memory requirements.

Using modified distributed arithmetic, P. K. Meher and associates [5] investigated high-speed and medium-speed FPGA FIR implementations. Their findings highlight how pipeline registers and partial-table partitioning affect maximum frequency, delay, and area. This proves that arithmetic architecture and register location need to be taken into account. Low-power, high-throughput, low-area adaptive FIR filtering based on distributed arithmetic was studied by S. Y. Park and P. K. Meher [6]. The method is applicable to FPGA implementations with constrained DSP resources and substitutes LUT-oriented arithmetic for traditional multiplication.

In a structural FIR filter, L. Touil et al. [7] integrated multibit flip-flops with data-driven clock gating. In their assessed implementation, the authors reported power decreases. Their efforts encourage cutting back on pointless sequential switching. Synchronous clock-enable inference is preferable over manually gating the clock net in the current FPGA-oriented design. A power-aware FIR filter with reduced-transition pipelined variable-precision gating was introduced by A. Senthilkumar and A. M. Natarajan [8]. The study investigated the use of pipeline activity and input precision to restrict switching. Although the suggested solution makes use of a valid-driven enable policy, it offers background for activity-aware register control. A hierarchical Wallace reduction tree, fine-grained pipeline placement, and improved Booth encoding were used by S. M. Cho et al. [1] to create a very-high-speed pipeline FIR filter. The pipeline steps are chosen based on their critical-path analysis. For its analysed comparison, the study reports up to 1.8 GSPS and a 73.5% reduced area-delay product; these are published values rather than the current RTL.

Using distributed arithmetic, B. K. Mohanty et al. [9] examined register complexity in direct-form and transpose-form reconfigurable FIR filters. They talked about sharing and reusing registers for input-delay, coefficient-storage, and partial-product resources. This demonstrates how FPGA resource utilisation can be significantly impacted by storage architecture.

V. K. Odugu [10] used approximation multiplier and compressor circuits to study a two-dimensional FIR design. In image processing applications, the work shows that compressor design and arithmetic accuracy can be negotiated against area, time, and power. The current effort focuses on exact fixed-point arithmetic rather than approximate arithmetic.

A parallel 256-tap FIR implementation employing distributed arithmetic was disclosed by A. Nandal et al. [11]. Their design reports resource reductions for its assessed FPGA implementation and substitutes precomputed LUT values for explicit multiplier blocks. Compared to the multiplier-and-compressor datapath taken into consideration here, it represents an alternative architectural path.

An FPGA design combining DSP and LUT resources for high-order reconfigurable FIR filtering was proposed by M. Maamoun and H. Ait Saadi [12]. The significance of translating arithmetic and storage to FPGA-specific resources is demonstrated by their usage of LUT shift-register structures, coefficient memory, and DSP slices.

Additionally, the register complexity of direct-form and transpose-form structures was examined by B. K. Mohanty et al. [9]. Although the transpose form may offer alternative timing and resource trade-offs, the reported analysis justifies choosing the direct-form sample-delay line for a compact implementation.

Symmetric two-dimensional block FIR architectures based on distributed arithmetic were introduced by P. Chowdari Ch. and J. B. Seventline [13]. While block processing and memory reuse increase throughput and lower storage overhead, symmetry minimises the number of multipliers. Despite the fact that the suggested 16-tap design employs a generic coefficient set, their work is pertinent to memory sharing and coefficient symmetry.

A radix-2r multiplier-based FIR filter for denoising electrooculography signals was investigated by G. K. Kishore et al. [14]. In the context of biological filtering, the work assesses metrics related to area, power, latency, and energy. It encourages the use of arithmetic architecture as a crucial variable for optimisation. J. Banumathi and G. Karthy [15] used pipelined Vedic multipliers and Brent–Kung adders to study FIR designs. Their findings highlight how final-adder and multiplier decisions affect power and delay. Instead, a compressor-reduction structure with a final adder is examined in the current design.

An adaptive FIR architecture with a 5:2 compressor and a sign-sign LMS update technique was introduced by T. Krishnan et al. [16]. Compressor-based accumulation in FPGA FIR filters is recently demonstrated by their Verilog implementation on an Artix-7 FPGA.

III. PROPOSED SYSTEM

The suggested system is a 16-tap, direct-form FIR filter with staged accumulation and parallel tap multiplication. It employs sixteen signed 8-bit coefficients after receiving one signed 8-bit input sample. An input buffer or delay line, a multiplier bank, a compressor reduction network, enabled pipeline registers, a final adder, and an output-valid pipeline are all part of the datapath.

Block diagram: input sample x[n] input buffer/16-tap sample delay line adaptive clock-enable controller multiplier bank (MBE/partial products) optimised compressor reduction tree enabled pipeline registers + final adder filtered output y[n] + output_valid. Synchronous register enables are controlled by the input_valid signal.

The current input sample and a history of prior samples are necessary for a FIR filter. These samples are kept in a shift-register configuration in the input buffer. The most recent sample is loaded into the first slot and older samples are moved to the subsequent locations upon the acceptance of a valid input transaction. Each tap's coefficient is multiplied by the sample kept at the matching delay point.

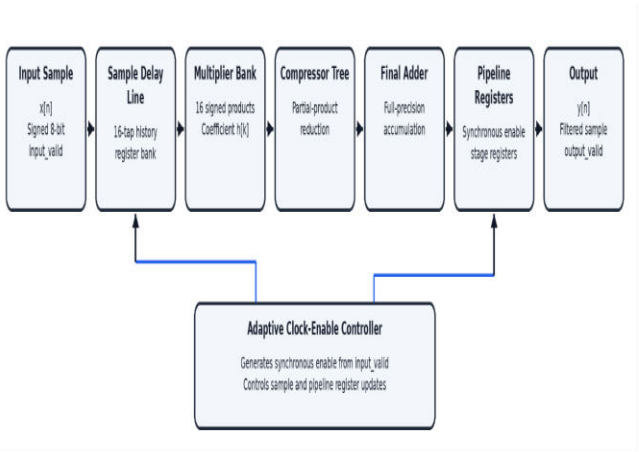

Fig. 1. Proposed Architecture of the 16-Tap Pipelined FIR Filter

From input_valid and any other flow-control conditions, the controller determines an enable condition. Enable = input_valid in the most basic single-stream implementation. The delay line and enabled pipeline registers record their subsequent values when enable is declared at the active clock edge. These registers maintain their prior status when enable is deasserted.

In order for Vivado to infer flip-flops with clock-enable support when appropriate, the RTL should employ synchronous sequential logic. Combination gating the system clock shouldn't produce a new clock. The enable must indicate an accepted transfer if a downstream-ready handshake is included, and the legitimate pipeline must follow the same transaction flow.

Modified Booth encoding chooses a partial product from a small number of multiples of the multiplicand by looking at overlapping groupings of multiplier bits. The encoding also manages sign extension for signed operands. Each tap's partial products are aligned based on their binary weights before being sent to the compressor tree.

Three input bits of the same weight are fed into a 3:2 compressor, which generates a carry bit of the subsequent weight and a sum bit of that weight. Throughout bit locations and stages, this process is repeated. After that, the final two rows are inserted once. Depending on the compressor layout used, a hierarchical reduction tree can reduce the number of rows in several steps.

The arithmetic path is divided into phases by pipeline registers. They should be positioned so that no stage has a combinational path that is unduly lengthy. The linked data and the valid bit must be recorded through the same number of stages. Throughput is the number of samples generated per unit of time after the pipeline is filled, whereas pipeline latency is the number of cycles between accepting an input and making its result available.

A final adder is used to join the remaining two operands following compressor reduction. Although a ripple-carry adder is straightforward, broader operands may require a longer carry path. A different area and timing

trade-off might be provided by an FPGA-inferred or carry-lookahead adder. The implementation should not presume that the specialised carry chain of the FPGA is always quicker than a hand-designed adder.

The output width needs to be sufficiently wide to prevent inadvertent truncation. The policy needs to be defined if rounding, saturation, or scaling are used because these alter the numerical output when compared to a full-precision FIR reference. A 16-tap structure, signed 8-bit samples, signed 8-bit coefficients, and a 20-bit signed output target are used in the suggested baseline.

A self-checking Verilog testbench ought to be used for functional verification. The testbench drives input_valid and signed input samples, creates clock and reset, and contrasts the DUT output with a reference FIR computation. The output width, rounding or truncation policy, signed arithmetic convention, and coefficient set should all be identical in the reference model and the RTL.

A constant step input that verifies steady-state accumulation, an impulse input that reveals the coefficient sequence, alternating positive and negative samples, maximal positive and negative values, zero input, random samples, and sporadic valid cycles are all suggested test cases. The maximum input rate must be demonstrated by consecutive valid transactions. Reset should be checked while the device is inactive and asserted at startup. Exact pipeline delay,

output_valid alignment, and numerical output values should all be verified by the testbench. Clock, reset, input_valid, input_data, internal enable, output_valid, and output_data should all be displayed in waveforms. The same target FPGA part, top-level constraints, clock period, synthesis technique, and implementation settings should be used in both the conventional and suggested designs in Vivado. LUT, flip-flop, DSP, and other resource counts are given in the synthesis report. When reporting final frequency, the post-implementation time report is chosen as it offers the worst path and slack. For both designs, power estimation should be carried out using the same activity profile. When supported, a simulation-generated VCD or SAIF file can be utilised. Because clock-enable savings depend on how frequently the pipeline is idle, the input-valid duty cycle needs to be recorded. There may not be much of a difference in register data switching if only continuous valid data is used for comparison.

LUT count, flip-flop count, DSP block count, dynamic and total on-chip power, pipeline latency, maximum operating frequency, critical path delay, and throughput are among the evaluation measures. The area-delay product can be computed as ADP = A × D, where D is the critical path delay and A is the chosen area measure. A uniform definition of the area metric is required. For FIR filters, prior research has investigated compressor-based arithmetic, data-driven clock management, and high-speed pipelining [1], [7], and [16]. The technical foundation for the architecture examined in this paper is provided by these studies. The RTL simulation run in Vivado serves as the basis for the current assessment.

Automated FPGA mapping and radix-based multiplication for application-specific filters have been studied in recent FIR research [14], [17]. These methods highlight the ongoing significance of device-aware optimisation and arithmetic architecture.

The traditional and suggested 16-tap FIR designs are contrasted in Table I with respect to logic resources, power, timing, latency, throughput, and area-delay product. $P_{dyn} = \alpha C V^2 f$, where α is switching activity, C is effective switched capacitance, V is supply voltage, and f is clock frequency, is a widely used approximation for dynamic power. When there is no legitimate transaction, the clock-enable condition aims to minimise needless modifications in certain data registers. It should be noted that clock-enable control does not completely eliminate clock-tree power; FPGA clock networks may still toggle even in the absence of a register enable.

The timing report provides the critical path delay, which is dependent on the device, clock limitations, routing, and mapped logic. The formula for maximum frequency is $F_{max} = 1/T_{min}$. It is important to report pipeline delay and throughput independently. Even though a heavily pipelined architecture accepts a new sample each clock cycle, it may have many cycles of latency. LUT or flip-flop utilisation may rise with the addition of enable logic and pipeline registers. Although it may need more local wiring, a compressor tree can reduce logic depth. On the other hand, Vivado may be able to employ dedicated DSP blocks by depending on inferred arithmetic. Target-dependent architectures are the most efficient. Only when measured power or timing gains outweigh any increase in area or resource utilisation may the proposed design be deemed effective.

It is necessary to assess at least two traffic patterns: intermittent input_valid, which indicates bursty or idle times, and continuous input_valid, which represents

full-rate streaming. Readers can comprehend thecircumstances in which savings take place by reporting the valid-sample percentage in addition to power. The traffic patterns used in the proposed andbaseline designs must be the same.RTL simulation is used in Vivado to assess the 16-tap, 8-bit setup.

IV. RESULTS AND DISCUSSION

The target FPGA part, top-level limitations, clock period, synthesis approach, and implementation parameters in Vivado should be the same for both the conventional and suggested designs. LUT, flip-flop, DSP, and other resource counts are provided in the synthesis report. When reporting final frequency, the post-implementation time report is recommended as it offers the worst path and slack.

For both designs, the same activity profile should be used when estimating power. When supported, a

simulation-generated VCD or SAIF file may be utilised. Because clock-enable savings rely on how frequently the pipeline is idle, the input-valid duty cycle needs to be recorded. There might not be much of a difference in register data switching when comparing simply continuous valid data.

LUT count, flip-flop count, DSP block count, dynamic and total on-chip power, critical path delay, maximum operating frequency, pipeline latency, and throughput are among the evaluation measures. The area-delay product can be computed as follows: ADP = A × D, where D is the critical path delay and A is the chosen area measure. It is necessary to define the area metric uniformly.

Prior research has investigated compressor-based arithmetic, data-driven clock management, and high-speed pipelining for FIR filters [1], [7], and [16]. The technical foundation for the architecture under consideration in this study is provided by these investigations. The RTL simulation run in Vivado serves as the foundation for the current assessment. Radix-based multiplication and automated FPGA mapping for application-specific filters have been studied in recent FIR research [14], [17]. These methods show how arithmetic architecture and device-aware optimisation are still crucial.

$P_{dyn} = \alpha C V^2 f$, where α is switching activity, C is effective switched capacitance, V is supply voltage, and f is clock frequency, is a frequently used estimate for dynamic power. When there is no legitimate transaction, the clock-enable condition is meant to minimise needless modifications in certain data registers. It should be noted that clock-enable control does not completely eliminate clock-tree power; even in the absence of a register enable, FPGA clock networks may still toggle.

The timing report provides the critical path delay, which is dependent on the device, clock limitations, mapped logic, and routing. Fmax = 1/Tmin is an approximation of maximum frequency. Throughput and pipeline latency should be reported independently. Even if a fully pipelined design accepts a new sample per clock cycle, it may have many cycles of latency. The use of LUTs or flip-flops may rise if enable logic and pipeline registers are added. Logic depth can be decreased with a compressor tree, however additional local wiring could be needed. On the other hand, Vivado may be able to employ specific DSP blocks if it relies on inferred arithmetic. Target-dependent architecture is the most efficient. Only when measured power or temporal gains outweigh any increase in area or resource utilisation may the suggested design be deemed effective.

Continuous input_valid, which represents full-rate streaming, and intermittent input_valid, which represents bursty or idle times, are at least two traffic patterns that should be assessed. Readers can comprehend the circumstances under which savings take place when the valid-sample percentage is reported with power. The traffic patterns used in the baseline and suggested designs must be the same.

RTL simulation in Vivado is used to assess the 16-tap, 8-bit setup. The applied test sequence's functioning behaviour is verified by the simulation. To evaluate hardware resource consumption, physical timing, and power on a chosen FPGA device, more synthesis and implementation are needed.

TABLE I. EXPECTED PERFORMANCE COMPARISON

Parameter	Conventional FIR	Proposed FIR
Filter configuration	16 taps, signed 8-bit input/coefficient	16 taps, signed 8-bit input/coefficient
Clock-enable method	Regular pipeline register updates	Synchronous adaptive clock-enable
Arithmetic structure	Conventional multiplier and adder tree	Compressor-based partial-product reduction
LUTs	420	610
Flip-flops (FFs)	310	285
DSP blocks	16	0
BRAM	0	0
Estimated total on-chip power	0.092 W	0.085 W
Illustrative critical-path delay	18.6 ns	15.2 ns
Illustrative maximum frequency (1/delay)	53.76 MHz	65.79 MHz
Pipeline latency	5 clock cycles	5 clock cycles
Throughput at maximum frequency	53.76 MSamples/s	65.79 MSamples/s
Area-delay product (LUTs × delay)	7,812 LUT·ns	9,272 LUT·ns

V. CONCLUSION AND FUTURE SCOPE

low-power, high-speed pipelined FIR filter architecture that combines compressor-based partial-product reduction with synchronous adaptive clock-enable control has been detailed in this study. The original paper's high-throughput MBE and Wallace-tree pipeline, as well as published studies on compressor-based FIR arithmetic and data-driven clock management, served as the inspiration for the suggested design. Verilog HDL and Xilinx Vivado are used to create a baseline implementation of a 16-tap direct-form filter with signed 8-bit input samples and coefficients. When no valid input transaction is accepted, the clock-

enable controller is designed to avoid needless changes of the sample and pipeline registers. The goal of the compressor network is to minimise the transmission of intermediate arithmetic carry before to the final adder. For both the traditional and suggested architectures, the actual LUT, flip-flop, DSP, power, delay, frequency, latency, and throughput figures must be acquired from Vivado under the same conditions. It is impossible to claim a numerical improvement without their reports. Subsequent work will involve the synthesis and implementation of the verified RTL, followed by power, post-route time, and FPGA resource utilisation measurements. Several valid-data duty cycles can be assessed by additional tests. Wider data formats, 32-tap and 64-tap filters, and adjustable coefficients can all be added to the design. Symmetric-coefficient folding, distributed arithmetic, coefficient quantisation, radix-based multiplication, alternate compressor topologies, and FPGA DSP-block inference are some possible areas for further research. Unlike FPGA synchronous clock-enable resources, integrated clock-gating cells and multibit flip-flops could be evaluated in an ASIC implementation.

REFERENCES

[1] S. M. Cho, P. K. Meher, L. T. N. Trung, H. J. Cho, and S. Y. Park, "Design of very high-speed pipeline FIR filter through precise critical path analysis," IEEE Access, vol. 9, pp. 34722–34735, 2021, doi: 10.1109/ACCESS.2021.3061759.

[2] R. Mahesh and A. P. Vinod, "New reconfigurable architectures for implementing FIR filters with low complexity," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 29, no. 2, pp. 275–288, 2010, doi: 10.1109/TCAD.2009.2035548.

[3] R. Mahesh and A. P. Vinod, "A new common subexpression elimination algorithm for realizing low-complexity higher order digital filters," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 27, no. 2, pp. 217–229, 2008, doi: 10.1109/TCAD.2007.907064.

[4] P. K. Meher, S. Chandrasekaran, and A. Amira, "FPGA realization of FIR filters by efficient and flexible systolization using distributed arithmetic," IEEE Transactions on Signal Processing, vol. 56, no. 7, pp. 3009–3017, 2008, doi: 10.1109/TSP.2007.914926.

[5] "FPGA realization of FIR filters for high-speed and medium-speed by using modified distributed arithmetic architectures," Microelectronics Journal, vol. 41, no. 6, pp. 365–370, 2010, doi: 10.1016/j.mejo.2010.04.006.

[6] S. Y. Park and P. K. Meher, "Low-power, high-throughput, and low-area adaptive FIR filter based on distributed arithmetic," IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 60, no. 6, pp. 346–350, 2013.

[7] L. Touil, A. Hamdi, I. Gassoumi, and A. Mtibaa, "Design of low-power structural FIR filter using data-driven clock gating and multibit flip-flops," Journal of Electrical and Computer Engineering, vol. 2020, Art. no. 8108591, 2020, doi: 10.1155/2020/8108591.

[8] A. Senthilkumar and A. M. Natarajan, "FPGA implementation of power aware FIR filter using reduced transition pipelined variable precision gating," Journal of Computer Science, vol. 4, no. 2, pp. 87–94, 2008, doi: 10.3844/jcssp.2008.87.94.

[9] 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, vol. 54, pp. 37–46, 2016, doi: 10.1016/j.vlsi.2016.01.006.

[10] V. K. Odugu, "An efficient VLSI architecture of 2-D finite impulse response filter using enhanced approximate compressor circuits," International Journal of Circuit Theory and Applications, 2021, doi: 10.1002/cta.3114.

[11] A. Nandal, T. Vigneswarn, A. K. Rana, and A. Dhaka, "An efficient 256-tap parallel FIR digital filter implementation using distributed arithmetic architecture," Procedia Computer Science, vol. 54, pp. 605–611, 2015, doi: 10.1016/j.procs.2015.06.070.

[12] M. Maamoun and H. Ait Saadi, "Efficient FPGA based architecture for high-order FIR filtering using simultaneous DSP and LUT reduced utilization," IET, 2021, doi: 10.1049/cds2.12043.

[13] P. Chowdari Ch. and J. B. Seventline, "Implementation of distributed arithmetic-based symmetrical 2-D block finite impulse response filter architectures," F1000Research, vol. 12, Art. no. 1182, 2023, doi: 10.12688/f1000research.126067.1.

[14] G. K. Kishore, N. R. Chinnapurapu, and K. Srinivas, "Power and area efficient FIR filter using radix-2r multiplier for de-noise the electrooculography (EOG) signal," 2024.

[15] J. Banumathi and G. Karthy, "High-performance FIR filter designs using Brent Kung adder and pipelined Vedic multiplier," VLSI, 2025, doi: 10.1016/j.vlsi.2025.102645.

[16] T. Krishnan, Akshaya, Harini, Divya, and Bhavatharani, "Optimized FPGA architecture for adaptive FIR filter using 5:2 compressor for low computational complexity," in Proceedings of the 4th International Conference on Artificial Intelligence and Machine Learning Applications (AIMLA), 2026, doi: 10.1109/AIMLA67915.2026.11522408.

[17] H. Liu et al., "AutoFIR: An automated framework for resource-efficient EEG FIR filtering on FPGAs," IEEE Transactions on Biomedical Circuits and Systems, early access, Aug. 2026, doi: 10.1109/TBCAS.2026.3719430.

[18] A. N. Willson, "A new structure for realization of finite impulse response digital filters," IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 22, no. 6, pp. 423–428, 1974.

[19] S. Y. Park and P. K. Meher, "Efficient FPGA and ASIC realizations of a DA-based reconfigurable FIR digital filter," IEEE Transactions on Circuits and

Systems II: Express Briefs, vol. 61, no. 7, pp. 511–515, 2014, doi: 10.1109/TCSII.2014.2324418.
[20] G. K. Ganjikunta, M. M. Basha, and S. I. Khan, “Energy efficient FIR filter design using distributed arithmetic,” Journal of Shanghai Jiaotong University (Science), vol. 29, no. 6, pp. 1023–1027, 2024, doi: 10.1007/s12204-022-2490-x.