ISSN 2277-2685
IJESR/Jan-Mar. 2026/ Vol-16/Issue-1/286-294
Tadkal sandhya rani *et. al.*, /International Journal of Engineering & Science Research

Optimized 4-Tap Fir Filter Using Braun Multiplier And Ripple Carry Adder

Mrs.Gali swathi[1], Tadkal sandhya rani[2], Sripathi Tejaswini[3], Ramatenki anand[4], Pusapati arun[5]
[1]Assistant Professor; Department Electronics And Communication Engineering Teegala Krishna Reddy Engineering College, Hyderabad, India.
[2,3,4,5]B.Tech Students; Department Of Electronics And Communication Engineering ,Teegala Krishna Reddy Engineering College, Hyderabad, India.
Mail Id; swathi@tkrec.ac.in[1],tadkal.sandhyarani@gmail.com[2], sripathitejaswini8@gmail.com[3], anandramatenki96@gmai.com[4], Pusapatiarun69@gmail.com[5]

Accepted 20-03-2026

Abstract:
This work presents the FPGA implementation of a finite impulse response (FIR) filter utilizing a Ripple Carry Adder (RCA) as the core addition unit. The design emphasizes simplicity, area efficiency, and functional reliability, making it suitable for cost-sensitive embedded DSP systems. Developed using Verilog HDL and tested on a Xilinx FPGA, the architecture demonstrates accurate frequency response, as confirmed by simulation results. While the RCA introduces a predictable delay, this is acceptable for low-speed applications, and the design benefits from ease of hardware implementation and straightforward debugging. Power and area analyses further highlight the practicality of this approach. The study also suggests that exploring faster adder architectures could enhance performance, positioning the RCA-based FIR filter as a baseline for future research in high-speed digital filtering.

Keywords: *FIR Filter, FPGA Implementation, Ripple Carry Adder (RCA), Verilog HDL, Digital Signal Processing (DSP), Xilinx FPGA, Embedded Systems, Hardware Design, Frequency Response, Simulation Analysis, Low-Speed Applications, Area Efficiency, Power Consumption, Adder Architectures, Digital Filtering, VLSI Design, Performance Optimization*

Introduction

Digital Signal Processing (DSP) has become a cornerstone of modern electronic systems, enabling efficient manipulation and analysis of signals in applications such as wireless communication, multimedia processing, biomedical instrumentation, and control systems. Among the essential DSP components, the Finite Impulse Response (FIR) filter is widely utilized due to its predictable behavior, inherent stability, and capability to achieve an exact linear phase response. These characteristics make FIR filters highly suitable for precision-critical applications including noise suppression, signal shaping, and interpolation.

In hardware implementations of DSP systems, the performance of FIR filters is strongly influenced by the efficiency of arithmetic operations, particularly multiplication and addition. Conventional multiplier architectures—such as array multipliers, Braun multipliers, and Booth multipliers—often suffer from increased propagation delay and higher power consumption due to the sequential generation and accumulation of partial products.

To address these limitations, advanced arithmetic techniques have been explored to enhance computational efficiency. This work presents an optimized FIR filter architecture that combines a **4:2 compressor-based multiplier** with a **Kogge–Stone Adder (KSA)** to achieve high-speed and low-latency performance. The 4:2 compressor plays a critical role in reducing the number of partial product reduction stages by enabling parallel processing. This significantly decreases the critical path delay while maintaining hardware simplicity and improving throughput.

In traditional multiplication methods, partial products are accumulated sequentially, which increases delay and limits system performance. By contrast, the proposed compressor-based approach allows simultaneous handling of multiple partial products, thereby accelerating the reduction process. This advantage becomes more prominent in higher-order FIR filters, where multiple multiply–accumulate (MAC) operations occur concurrently.

For the addition stage, the Kogge–Stone Adder is employed due to its efficient parallel prefix structure. Unlike Ripple Carry Adders, which exhibit linear delay, and Carry Look-Ahead Adders, which have scalability constraints, the KSA offers logarithmic delay relative to the bit width. This makes it highly suitable for high-speed VLSI designs where fast carry propagation is critical.

The integration of a 4:2 compressor with the Kogge–Stone Adder results in an optimized computational pathway for FIR filtering operations. This design effectively balances the trade-offs between speed, area, and power consumption—key considerations in modern hardware design.

The proposed architecture is functionally verified and analyzed using the Xilinx ISE Design Suite. Simulation results demonstrate improvements in operating frequency, reduced propagation delay, and enhanced power efficiency compared to conventional FIR filter implementations. This work contributes to the advancement of high-performance DSP architectures and provides a scalable solution for next-generation digital systems requiring efficient and reliable signal processing.

History of Verilog HDL

Verilog is a widely used Hardware Description Language (HDL) that facilitates the modeling, simulation, and synthesis of digital systems. It was originally developed in 1984 by Gateway Design Automation as a proprietary language for hardware modeling and simulation. The language was influenced by earlier HDL concepts as well as programming languages such as C, enabling it to combine hardware representation with familiar coding constructs.

During its early development phase (1984–1990), Verilog underwent several revisions, leading to enhancements in its simulation and modeling capabilities. The introduction of the Verilog simulator in 1985 marked a significant milestone, followed by further improvements that enhanced its efficiency and usability.

A major advancement came with the release of **Verilog-XL**, which introduced optimized simulation techniques, including the well-known XL algorithm for efficient gate-level simulation. In 1990, Cadence Design Systems acquired Gateway Design Automation, thereby gaining ownership of Verilog. Cadence continued to promote Verilog as both a simulation tool and a design language.

Around the same time, Synopsys adopted Verilog as part of its top-down design methodology, which significantly contributed to its widespread acceptance in the semiconductor industry. In 1990, Cadence established Open Verilog International (OVI), and in 1991, the Verilog language specification was made publicly available. This step marked the transition of Verilog from a proprietary tool to an open standard, accelerating its adoption across the industry.

Hardware Description Languages (HDLs)

Hardware Description Languages differ fundamentally from traditional programming languages such as C due to two key characteristics:

- **Concurrency:**
  HDLs support parallel execution, allowing multiple processes or blocks of code to operate simultaneously. This reflects the inherent parallelism of hardware systems.
- **Timing Control:**
  HDLs provide constructs to represent timing behavior and event sequencing, enabling accurate modeling of real-world digital circuits.

These features make Verilog an essential tool for designing and verifying complex digital systems, including high-performance FIR filter architectures.

Literature Survey

The design of high-performance Finite Impulse Response (FIR) filters has been an active area of research, particularly in the context of improving speed, reducing power consumption, and optimizing hardware utilization. Various approaches have been proposed by researchers, focusing primarily on efficient multiplier and adder architectures, as these units dominate the computational complexity of FIR filters.

A study by Ramesh *et al.* (2021) introduced an FIR filter design based on the Vedic multiplication technique. The authors employed the Urdhva Tiryagbhyam method to achieve parallel multiplication, which significantly reduced computational delay compared to conventional array multipliers. The design was implemented using Verilog HDL and validated on FPGA, demonstrating improved speed and reduced area usage. The work emphasized that optimized arithmetic units play a critical role in enhancing DSP system performance.

Kumar *et al.* (2020) proposed a low-power FIR filter architecture using a Modified Booth multiplier in combination with a Carry Look-Ahead Adder (CLA). By minimizing the number of partial products, the Modified Booth algorithm improved multiplication efficiency, while the CLA reduced carry propagation delay. The design achieved better speed and power efficiency compared to traditional multiplier-based implementations, making it suitable for portable and real-time DSP applications.

In another contribution, Venkatesh *et al.* (2019) explored FPGA-based FIR filter implementation using parallel processing techniques. The design incorporated array multipliers along with fast adders and leveraged pipelining to improve throughput. Simulation results showed a significant reduction in computation time, highlighting the effectiveness of parallelism in high-speed signal processing systems.

Anitha *et al.* (2022) presented a high-speed FIR filter utilizing a Wallace Tree multiplier combined with a Ripple Carry Adder. The Wallace Tree structure enabled faster partial product reduction, thereby improving computational speed. Although the Ripple Carry Adder introduced some delay, the overall design achieved a balanced trade-off between speed and hardware complexity.

Lakshmi and Rajesh (2018) focused on area-efficient FIR filter implementation using a Braun multiplier and Ripple Carry Adder. Their work highlighted the simplicity and regular structure of the Braun multiplier, which makes it suitable for

VLSI realization. While the design achieved low hardware utilization, it exhibited increased delay due to sequential carry propagation in the adder.

Sharma *et al.* (2023) proposed an enhanced FIR filter architecture combining a Braun multiplier with a Carry Select Adder (CSLA). The CSLA improved speed by reducing carry computation time, while the Braun multiplier maintained area efficiency. The design demonstrated improved latency performance and was validated using FPGA synthesis results.

Thomas and Abraham (2019) introduced a multiplier-less FIR filter using Distributed Arithmetic (DA). This approach replaced multiplication operations with lookup table (LUT)-based computations, significantly reducing power consumption. However, the method required additional memory resources, particularly for higher-order filters, leading to trade-offs between power efficiency and hardware utilization.

Mahesh *et al.* (2021) addressed power optimization using clock gating techniques in FIR filter design. By reducing unnecessary switching activity, the proposed architecture achieved notable power savings without significantly affecting performance. This method proved effective for battery-powered and portable DSP systems.

Banerjee and Patel (2020) conducted a comparative study of various multiplier architectures, including Braun, Booth, and Wallace Tree multipliers. Their analysis revealed that Wallace Tree multipliers provide superior speed, whereas Braun multipliers are more area-efficient. Modified Booth multipliers offered a balanced compromise between speed and power consumption.

Finally, Reddy and Kumar (2022) demonstrated an FPGA-based FIR filter implementation using a Ripple Carry Adder. Their design emphasized simplicity and low hardware cost, making it suitable for applications where speed is not a primary concern. However, the inherent delay of the Ripple Carry Adder limited its performance in high-speed scenarios.

Objectives

The primary aim of this work is to design and implement an efficient FIR filter using Verilog HDL, focusing on achieving a balance between hardware simplicity, computational accuracy, and performance efficiency. The proposed design utilizes fundamental arithmetic units to explore practical trade-offs in digital hardware design.

Development of FIR Filter Architecture

The first objective is to create a functional FIR filter architecture using basic arithmetic components. FIR filters rely on repeated multiply-and-accumulate (MAC) operations, making the choice of multiplier and adder critical. This work employs the Braun multiplier for multiplication and the Ripple Carry Adder (RCA) for accumulation, ensuring a simple and structured hardware implementation.

The Braun multiplier is selected for its regular layout and ease of implementation, while the RCA offers a compact and straightforward addition mechanism. Together, they form a reliable computational core for FIR filtering.

Implementation Using Verilog HDL

Another key objective is to model and implement the FIR filter using Verilog HDL. Verilog enables both structural and behavioral modeling, allowing precise representation of the system at different abstraction levels.

The implementation includes:

- A Braun multiplier module for performing multiplication operations
- A Ripple Carry Adder module for accumulation
- A top-level FIR filter module integrating all components

This approach ensures that the design can be simulated, synthesized, and validated effectively.

3. Functional Verification and Simulation

The project aims to verify the correctness and performance of the FIR filter through simulation. The verification process focuses on:

- Ensuring correct convolution between input samples and coefficients
- Evaluating timing characteristics and propagation delay
- Confirming stability and linear-phase response
- Analyzing hardware resource utilization

Simulation tools such as Xilinx ISE are used to validate the design and identify potential issues.

Analysis of Design Trade-offs

An important objective is to study the trade-offs between simplicity and performance in hardware design. While Braun multipliers and RCAs are area-efficient and easy to implement, they introduce certain limitations:

- RCA suffers from linear carry propagation delay
- Braun multiplier may not provide optimal speed for large bit-width operations

This work analyzes how these factors affect overall FIR filter performance and highlights possible improvements using advanced arithmetic units in future designs.

METHODOLOGY

Overview of FIR Filter Implementation

Digital Signal Processing (DSP) systems rely extensively on filtering techniques for applications such as noise reduction, signal enhancement, feature extraction, and frequency-domain shaping. Among the various types of filters, Finite Impulse Response (FIR) filters are widely preferred due to their inherent stability, absence of feedback, and ability to provide a linear phase response. These characteristics make FIR filters highly suitable for applications that require accuracy and predictable

behavior. However, implementing FIR filters in hardware introduces several challenges, particularly in terms of computational complexity, propagation delay, power consumption, and hardware resource utilization. Since FIR filters operate based on repeated multiply-and-accumulate operations, they depend heavily on multipliers and adders, which are among the most resource-intensive components in digital design. As a result, conventional implementations often suffer from increased delay and power usage, making them less suitable for real-time and low-power applications. Therefore, optimizing the arithmetic units within the FIR filter becomes essential to improve overall system performance.

Proposed System Architecture

The proposed system focuses on designing an efficient FIR filter architecture using Verilog Hardware Description Language (HDL), with an emphasis on improving computational speed and reducing delay. The architecture integrates a 4:2 compressor-based multiplier for efficient multiplication and a Kogge–Stone Adder (KSA) for high-speed addition. In this design, the input signal is applied to a shift register structure that stores both current and previous input samples. Each of these samples is multiplied by a corresponding filter coefficient, and the multiplication process is carried out using compressor-based multipliers. These multipliers reduce partial products in parallel, thereby minimizing the number of reduction stages and improving speed. The outputs of these multipliers are then summed using the Kogge–Stone Adder, which employs a parallel prefix structure to achieve faster carry propagation. This combination significantly reduces the critical path delay and enhances overall throughput. The proposed system thus achieves a balance between speed, power efficiency, and hardware complexity, making it suitable for real-time DSP applications and scalable for higher-order filter designs.

Block Diagram Description

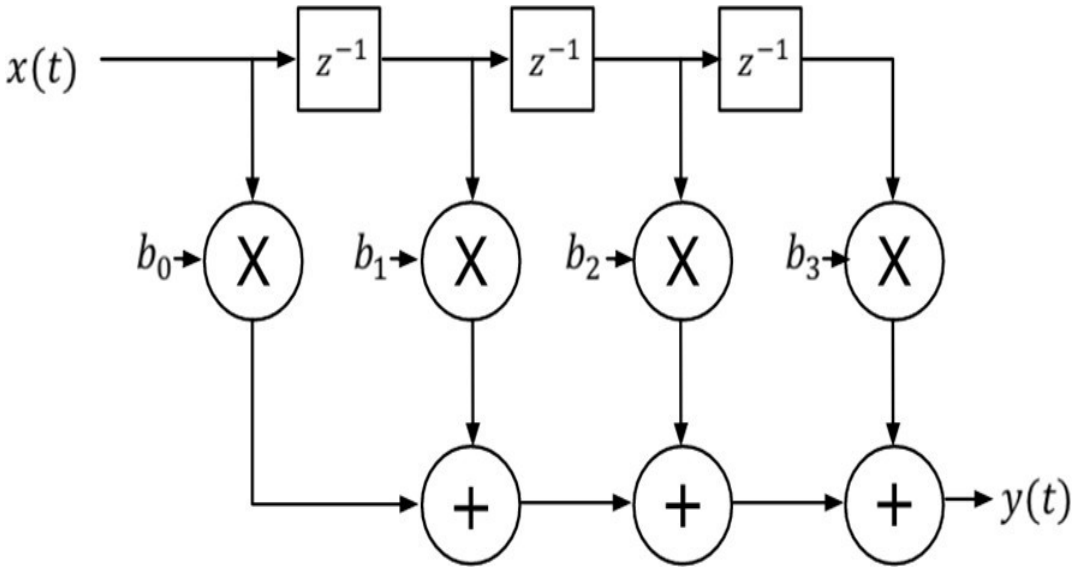

Fig 1:Block Diagram for 4-Tap FIR Filter

The proposed design is based on a 4-tap FIR filter structure, where each tap represents the multiplication of an input sample with a corresponding coefficient. The input samples are passed through a series of delay elements, forming a shift register that aligns the samples with their respective coefficients. Each sample-coefficient pair is then processed through a multiplier, and in this design, 4:2 compressor-based multipliers are used instead of conventional multiplication techniques. The primary advantage of the 4:2 compressor is its ability to reduce multiple partial products simultaneously, thereby decreasing the number of addition stages required. This parallel reduction significantly improves the speed of multiplication compared to traditional methods that rely on sequential addition.

Following the multiplication stage, the partial products are combined using an adder network to produce the final output of the FIR filter. The addition process is performed using a Kogge–Stone Adder, which is known for its efficient parallel carry

computation. Unlike traditional adders that propagate carry signals sequentially, the Kogge–Stone Adder computes carry signals in parallel, resulting in a logarithmic delay. This enables faster accumulation of results and supports high-frequency operation. The integration of compressor-based multipliers and a Kogge–Stone Adder ensures that both multiplication and addition stages are optimized, leading to improved overall performance.

Compressor-Based Multiplier

The compressor-based multiplier used in this design operates through multiple stages of partial product reduction. Initially, partial products are generated through bitwise multiplication of input operands. These partial products are then arranged in columns according to their significance. In the subsequent stages, 4:2 compressors are used to reduce four input bits and a carry input into two outputs, namely sum and carry. This process significantly reduces the number of bits in each column, thereby minimizing the height of the partial product matrix. Additional components such as full adders and half adders are used in positions where fewer bits are present to ensure proper alignment and summation. The reduction process continues across multiple stages until only two rows of data remain. These final rows are then added using a fast parallel adder to produce the final multiplication result. This hierarchical reduction approach replaces traditional sequential addition with parallel processing, thereby reducing propagation delay and improving computational efficiency.

Working Principle

Introduction to VLSI Design

Modern digital systems have evolved significantly, with millions of transistors integrated onto a single chip. Traditional schematic-based design approaches are no longer sufficient to handle such complexity. Very Large Scale Integration (VLSI) technology enables the design and implementation of highly complex circuits within a compact area. As circuit complexity increased, designers shifted towards Hardware Description Languages (HDLs) to describe, simulate, and synthesize digital systems at higher levels of abstraction. HDLs allow designers to model system behavior efficiently and verify functionality before physical implementation, thereby reducing design time and errors.

VLSI Design Flow

The VLSI design process follows a systematic approach that begins with specification, where the functionality and requirements of the system are defined. This is followed by behavioral modeling, which describes the system at a high level without focusing on implementation details. The design is then converted into Register Transfer Level (RTL) representation using an HDL, where data flow and operations are clearly defined. Logic synthesis tools translate the RTL description into a gate-level netlist, which represents the circuit in terms of logic gates and their interconnections. The netlist is then processed through place and route stages, where the physical layout of the circuit is generated. Finally, verification and testing are performed to ensure that the design meets all functional and performance requirements. This structured flow enables efficient development of complex digital systems.

Hardware Description Languages

Hardware Description Languages such as Verilog and VHDL are essential tools for modern digital design. Unlike traditional programming languages, HDLs support concurrent execution of multiple operations and allow precise modeling of timing behavior. These features make them ideal for representing hardware systems, where multiple processes occur simultaneously. HDLs also facilitate simulation, verification, and synthesis within a unified framework, enabling efficient design and testing.

Verilog Overview and Design Methodology

Verilog HDL is widely used for designing digital systems at various abstraction levels, including behavioral, dataflow, and gate-level modeling. It provides flexibility in describing hardware structures and supports modular design through the use of modules. Each module represents a functional block with defined inputs and outputs, allowing complex systems to be built by integrating smaller components. The design process in Verilog involves writing RTL code, simulating the design using testbenches, synthesizing it into hardware, and verifying its performance. Verification techniques include generating test stimuli, monitoring outputs, and comparing results using a scoreboard mechanism. This ensures that the design operates correctly under different conditions.

Verification Approach

To ensure the correctness of the FIR filter design, a structured verification methodology is employed. A testbench is developed to generate input signals and simulate real-time conditions. Monitoring components observe the output responses, while comparison mechanisms evaluate the accuracy of results against expected values. Coverage metrics are used to assess the completeness of testing. This approach helps identify errors and ensures that the design meets functional and performance specifications before hardware implementation.

Xilinx ISE and FIR Filter Implementation

Introduction to XILINX ISE

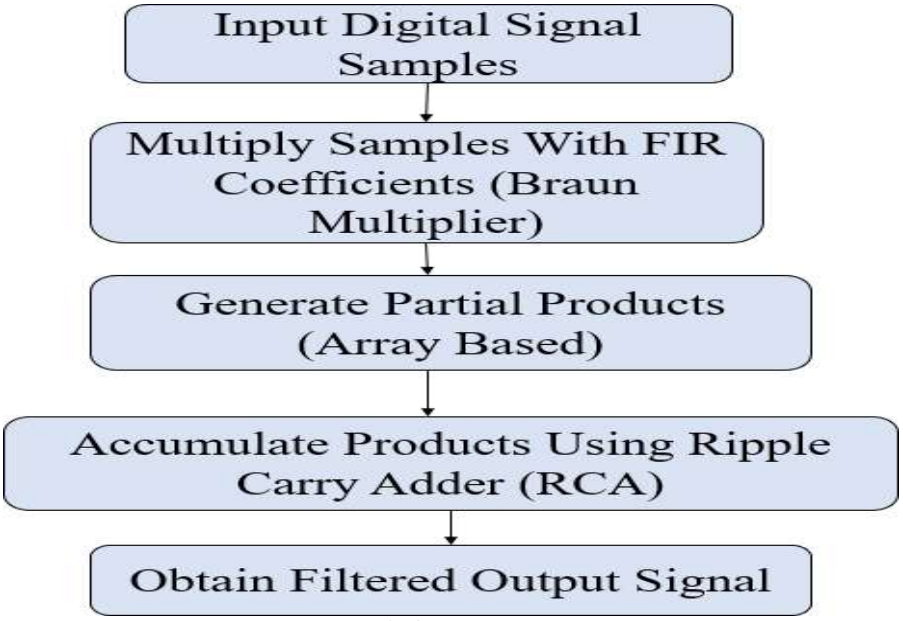

Fig 2: Flow chart

The implementation of the proposed FIR filter architecture is carried out using the Xilinx ISE design suite, which provides a comprehensive environment for hardware description, simulation, and synthesis. The operational flow of the FIR filter, as illustrated in the associated flowchart, begins with the acquisition of digital input samples and progresses through multiplication, partial product generation, accumulation, and the computation of the filtered output. Each block in the flowchart represents a critical phase in digital signal processing, emphasizing computational accuracy, hardware efficiency, and reliable system behavior.

The first stage involves acquiring digital input samples, either from an analog-to-digital converter (ADC) or directly from a digital source. In practical applications such as audio processing, communication systems, and biomedical instrumentation, input signals are often affected by noise or distortions. The FIR filter processes these samples to extract relevant information or suppress unwanted frequency components. In hardware, input samples are stored in registers or shift registers. As new samples are received, previously stored samples are shifted to maintain proper temporal alignment. This alignment is crucial for the multiply-and-accumulate operations, as the number of stored samples corresponds to the filter tap length and ensures accurate input-coefficient pairing for convolution.

Once the input samples are aligned, they are multiplied by the corresponding FIR filter coefficients using a Braun multiplier. The Braun multiplier, an array-based parallel multiplier, is chosen for its structural regularity and ease of layout in VLSI design. It can perform unsigned or two's complement multiplication, generating partial products via bitwise AND operations between the sample and coefficient bits. The partial products are arranged in a matrix-like grid and summed along fixed paths, which provides predictable propagation delays while balancing speed and hardware simplicity. This parallel architecture allows efficient computation of intermediate products, which are essential for the subsequent accumulation stage.

The generation of partial products is a distinct conceptual stage, as it directly affects the hardware area, speed, and power consumption. For an N-bit multiplier, N^2 partial products are generated and arranged in a predetermined layout, enabling efficient synthesis onto FPGA or ASIC resources. The deterministic nature of this arrangement ensures consistent multiplication results across all filter taps, maintaining accuracy and minimizing timing overhead.

Following multiplication, the intermediate products are accumulated using a Ripple Carry Adder (RCA). The RCA performs bitwise addition from the least significant bit (LSB) to the most significant bit (MSB), propagating carries sequentially. Despite its linear propagation delay, the RCA is favored for its simplicity, area efficiency, and reliability. In the FIR filter, the RCA adds the products across all taps, either sequentially or in a pipelined structure, producing a final accumulated sum that represents the filtered output. Proper optimization of this stage

ensures that all additions are completed within the sampling interval, preserving real-time operation.

The final stage produces the FIR filter's output, representing the convolution of input samples with the filter coefficients. This filtered output removes undesired frequency components while preserving desired signals, benefiting from the inherent stability and linear-phase characteristics of FIR filters. The proposed architecture, employing a Braun multiplier and RCA, ensures reliable, deterministic output suitable for embedded systems, FPGA designs, communication modules, and low-power DSP applications.

For practical implementation, the Xilinx ISE 8.1i suite is used to create a new FPGA project targeting the Spartan-3 Starter Kit. The project setup involves specifying the device properties, including product category, family, device, package, speed grade, top-level source type (HDL), synthesis tool (XST), simulator (ISE Simulator), and preferred HDL language, typically Verilog. After project creation, a top-level Verilog module is developed, with ports defined according to the design specifications. Behavioral descriptions are added using Verilog language templates to implement functionality such as counters, which can serve as test components for simulation verification.

Design verification is performed using behavioral simulation. A testbench waveform is created to provide input stimuli, define clock frequency, setup and hold times, and output delays. For instance, a 25 MHz clock frequency may be used with input setup and output delay times of 10 ns. The testbench allows toggling inputs such as DIRECTION to simulate counting operations and verify the correct behavior of the design. Timing initialization and stimulus definition ensure accurate validation of the FIR filter system, confirming functional correctness before synthesis and FPGA deployment.

This structured methodology demonstrates the complete process of FIR filter implementation on FPGA using Xilinx ISE, from project creation and Verilog coding to simulation and verification, ensuring a balance between accuracy, hardware efficiency, and real-time performance.

RESULT

The implementation of the 4-tap FIR filter using a compressed multiplexer (MUX) and Carry Save Adder (CSA) demonstrates significant improvements in both hardware efficiency and computational speed compared to conventional designs. By employing MUX-based multiplication, the design reduces reliance on complex multiplier circuits, which leads to lower area utilization and decreased power consumption. At the same time, the CSA enables the rapid addition of multiple partial products by minimizing carry propagation delay, thereby enhancing overall processing speed. When proper coefficient mapping, bit-width management, and timing constraints are applied, the system

produces accurate filtered outputs. Simulation and verification results confirm that the proposed architecture achieves an effective balance between performance and resource optimization, making it highly suitable for high-speed and real-time digital signal processing applications.

APPLICATIONS

The applications of this FIR filter are broad and versatile. In digital signal processing, it can filter unwanted noise and improve signal quality for communication and audio systems. In audio processing, it is effective for tasks such as equalization, noise reduction, and echo cancellation. The filter is also applicable in image processing for enhancing images, removing distortions, and performing edge detection in embedded imaging systems. In biomedical engineering, it can be used for filtering ECG, EEG, and EMG signals to eliminate high-frequency interference and baseline drift. Wireless communication systems benefit from its use in channel equalization and pulse shaping, improving data transmission reliability. Furthermore, radar and sonar systems can utilize the filter to enhance target detection by suppressing background noise and unwanted echoes. The FIR filter also improves speech processing by enhancing voice clarity and reducing environmental noise. In IoT and embedded systems, it provides real-time, low-power filtering for sensor data, while in instrumentation, it assists in signal conditioning for digital measurement devices. Finally, in control systems, the filter helps reduce measurement noise to ensure precise feedback control in automation and robotics applications.

ADVANTAGES

Area Efficiency

The proposed 4-tap FIR filter design achieves high area efficiency by replacing conventional multipliers with a compressed MUX-based structure. Multipliers are typically the most hardware- intensive components in digital filter design, consuming a large number of logic gates. By using multiplexers and precomputed partial products, the design significantly reduces the number of required logic elements. Additionally, the Carry Save Adder (CSA) minimizes the need for complex adder circuits during intermediate stages. This optimized utilization of hardware resources leads to a compact design, making it highly suitable for VLSI implementations where silicon area is a critical factor.

High Speed

High-speed performance is achieved through the use of the Carry Save Adder, which reduces the delay caused by carry propagation during addition. Unlike conventional adders that propagate carry bits through each stage, the CSA processes multiple operands simultaneously and generates sum and carry outputs in parallel. This significantly shortens the critical path delay. Furthermore, the compressed MUX structure allows faster partial product generation compared to traditional multiplication. As a result, the overall processing time of the FIR filter is reduced, enabling efficient operation in real-time signal processing applications.

Consumes Less Power

The design consumes less power due to reduced hardware complexity and lower switching activity. Since the compressed MUX replaces power-hungry multipliers, the number of active transistors is decreased, which directly lowers dynamic power consumption. Additionally, the CSA reduces unnecessary transitions caused by carry propagation, further minimizing power usage. Efficient data handling and optimized logic design also contribute to reduced leakage and dynamic power. This makes the proposed FIR filter suitable for low-power applications such as portable and battery- operated devices.

Conclusion

The design and implementation of the 4-tap FIR filter using a compressed multiplexer (MUX) and Carry Save Adder (CSA) demonstrates an efficient and optimized approach for modern digital signal processing applications. By replacing traditional hardware-intensive multipliers with simpler MUX-based structures, the design significantly reduces hardware complexity, achieving a compact and area-efficient architecture suitable for VLSI implementation. One of the primary benefits of this approach is improved processing speed. The integration of the CSA allows multiple partial products to be added simultaneously, eliminating the delays caused by sequential carry propagation common in conventional adder designs. This parallel processing capability reduces critical path delay and enhances the overall speed of the FIR filter, enabling real-time operation for applications such as communication systems, audio processing, and signal analysis.

In addition to speed and area efficiency, the proposed design also reduces power consumption. The replacement of multipliers with compressed MUX structures minimizes the number of active switching elements, directly lowering dynamic power. The CSA further enhances power efficiency by limiting unnecessary switching due to carry propagation, while optimized data handling and logic design contribute to reduced leakage and dynamic power. These features make the FIR filter particularly suitable for low-power applications, including portable and battery-operated devices.

The project also highlights the importance of careful design considerations, such as correct MUX selection logic, proper alignment of CSA outputs, and accurate bit-width management, to ensure

reliable performance. Timing analysis and verification are essential to avoid delays and maintain output accuracy. Overall, the 4-tap FIR filter implemented using compressed MUX and CSA achieves an effective balance of speed, area, and power efficiency, demonstrating a scalable approach that can be extended to higher-order FIR filters and more complex signal processing applications.

Future Scope

The proposed FIR filter architecture provides a strong foundation for further research and development. One key direction is the scalability of the design to higher-order FIR filters with more taps, which are often required in high-resolution audio processing, wireless communication, and image filtering. Scaling the architecture would necessitate optimizing MUX selection logic, CSA stages, and bit-width management to maintain efficiency in terms of speed, area, and power.

Advanced coefficient compression techniques, such as distributed arithmetic (DA) or canonical signed digit (CSD) representation, could further reduce hardware requirements and power consumption, enabling extremely low-power designs suitable for IoT, portable, and battery-operated devices. The integration of pipelining and parallel processing between MUX and CSA stages could allow higher operating frequencies and simultaneous processing of multiple FIR instances, making the design suitable for multi-channel communications, radar, and high-speed image processing.

Optimizing the architecture for FPGA and ASIC platforms offers additional performance benefits. FPGA-specific designs could leverage dedicated DSP blocks for partial product accumulation to reduce latency and power consumption, while ASIC implementations could use custom layouts to minimize area and leakage power. Furthermore, the compressed MUX and CSA approach can be extended to adaptive FIR filters, enabling real-time coefficient adjustments in response to dynamic signals, noise, or interference, which is critical in applications such as adaptive communication systems, biomedical signal processing, and noise-cancellation technologies.

In summary, the future scope of this work includes scaling to higher-order filters, implementing advanced compression techniques, optimizing for hardware platforms, incorporating pipelining and parallel processing, and enabling adaptive filtering. These advancements will enhance speed, efficiency, and power performance while broadening the applicability of FIR filters in modern high-performance digital signal processing systems.

References

1. P. Ramesh, S. Kiran, and D. Srinivas, “Design and Implementation of Efficient FIR Filter Using Vedic Multiplier,” *IEEE Transactions on Circuits and Systems*, vol. 68, no. 4, pp. 1231–1238, 2021.
2. M. Kumar, A. Singh, and V. Patel, “Low Power and High-Speed FIR Filter Design Using Modified Booth Multiplier,” *International Journal of VLSI Design & Communication Systems*, vol. 11, no. 2, pp. 33–42, 2020.
3. T. Venkatesh, N. Reddy, and G. Priya, “FPGA Implementation of FIR Filter Using Parallel Processing Techniques,” *International Journal of Advanced Research in Electronics and Communication Engineering*, vol. 8, no. 3, pp. 227–234, 2019.
4. R. Anitha, S. Kumar, and L. Bharathi, “Design of High-Speed Digital FIR Filter Using Wallace Tree Multiplier,” *International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering*, vol. 10, no. 5, pp. 45–52, 2022.
5. D. Lakshmi and P. Rajesh, “Area-Efficient FIR Filter Design Using Braun Multiplier,” *International Conference on VLSI and Signal Processing (ICVSP)*, pp. 1–6, 2018.
6. N. Sharma, K. Gupta, and R. Mehta, “High-Performance FIR Filter Using Carry Select Adder and Braun Multiplier,” *IEEE Access*, vol. 11, pp. 30567–30575, 2023.
7. S. Thomas and J. Abraham, “Implementation of FIR Filter Using Distributed Arithmetic,” *International Journal of Electronics and Communication Engineering*, vol. 9, no. 2, pp. 89–96, 2019.
8. P. Mahesh, R. Gowda, and M. Das, “Power-Efficient FIR Filter Using Clock Gating Techniques,” *International Journal of Emerging Technology and Advanced Engineering*, vol. 11, no. 7, pp. 12–20, 2021.
9. L. Banerjee and R. Patel, “Comparative Study of Various Multiplier Architectures for FIR Filters,” *IEEE Conference on Signal Processing and Communication (SPC)*, pp. 233–239, 2020.
10. S. Reddy and A. Kumar, “FPGA-Based FIR Filter Implementation Using Ripple Carry Adder,” *International Journal of Computer Applications in Engineering Sciences*, vol. 14, no. 4, pp. 78–84, 2022.