International Journal of Multidisciplinary Engineering in Current Research - IJMEC
Volume 10, Issue 6, June-2025, http://ijmec.com/, ISSN: 2456-4265

Efficient 3-Parallel Polyphase Odd Length FIR Filter using Brent Kung Adder and Booth Multiplier

Dr K Ashok Kumar, S Neeharika Jaiswal, P Nanditha, S Ruchitha

[1]Associate Professor, Department Of Ece, Bhoj Reddy Engineering College For Women, India.

[2,3,4]B. Tech Students, Department Of Ece, Bhoj Reddy Engineering College For Women, India.

ABSTRACT

In plan and execution of DSP processors decrease in power utilization and region advancement comprises the essential standards. Parallel Finite Impulse Response (FIR) filter is at the centre of the plan also, execution. In this paper, utilization of FFA based 3- equal polyphase FIR filter with streamlined adder and multiplier instead of customary adder and multiplier has been introduced. Accordingly, FFA based 3-parallel polyphase odd length FIR filter utilizing two distinct multipliers to be specific Vedic multiplier and Booth multiplier and the three distinct adders to be specific Ripple carry adder, carry lookahead adder, and Brent Kung adder has been proposed. FFA based 3-equal polyphase FIR filter have been additionally executed utilizing various multipliers and adders and are then thought about for different parameters. The addition is both in delay and area. Also, low power utilization also, defer decrease in the Booth multiplier as well as Brent Kung adder make it exceptionally ideal for planning of the parallel FIR filter for low power and little chip region VLSI applications.

1-INTRODUCTION

Digital Signal Processing (DSP) plays a pivotal role in various applications, including communications, audio processing, and image processing. FIR filters are fundamental components in DSP systems, responsible for tasks such as signal filtering and noise reduction. The efficiency of FIR filters is crucial for real-time processing, prompting continuous research into novel architectures to enhance their performance. This project introduces a High-Speed VLSI Implementation of an odd-length FIR Filter, leveraging advanced arithmetic units for optimized speed and area utilization.

The design incorporates a Brent Kung adder and a Booth multiplier, both recognized for their efficiency in high-speed arithmetic operations. The Brent Kung adder is a parallel prefix adder known for its regular structure and efficient carry propagation mechanism. It enables fast addition of multiple operands simultaneously, contributing to the overall speed enhancement of the FIR filter. The Booth multiplier is chosen for its ability to reduce the number of partial products generated during the multiplication process, resulting in a more streamlined and faster multiplication operation. This is particularly beneficial in FIR filters where a significant portion of computation involves multiplication. The project involves a comprehensive exploration of the architectural design, Verilog coding, and subsequent synthesis and optimization steps. The proposed FIR filter is implemented on a VLSI platform, emphasizing the use of modern technologies for efficient resource utilization and high-speed processing.

The methodology includes the selection of appropriate filter specifications, the design of individual components, and the integration of the Brent Kung adder and Booth multiplier into the FIR filter structure. The performance of the design is

rigorously evaluated through simulations and post-layout analyses. This research contributes to the field of VLSI-based FIR filter design by combining the strengths of a high-speed adder and multiplier. The ensuing benefits include reduced computation time, making it suitable for applications requiring rapid and efficient signal processing. The subsequent sections of this document will delve into the detailed design, implementation, and performance analysis of the proposed High-Speed FIR filter.

2-LITERATURE SURVEY

- Design and implementation of area efficient 2-parallel filters on FPGA using image system

  Parallel FIR filter is mostly used among various types of filters in Digital Signal Processing (DSP). This paper shows the design of area-efficient 2-parallel FIR filter using VHDL and its implementation on FPGA using image system. This paper gives the details basic blocks of area-efficient 2-parallel FIR digital filter. In this paper proposed 2-parallel digital FIR filter and area-efficient 2-parallel FIR filter are explained. Its simulation using Xilinx 14.2 are also discussed. It also presents the FPGA implementation of primary 2-parallel filter and area-efficient 2-parallel on Xilinx 14.2 Spartan 3E Starter Board XC3S500E chips and its results. Since adders are light weight in silicon area when compare with the multipliers, therefore multipliers are replaced by the adder to reduce area and delay of the parallel FIR filter. Xilinx ISE is used for simulating the design of the filter.
- Short-length FIR filters and their use in fast non-recursive filtering

  This paper provides the basic tools required for an efficient use of the recently proposed fast FIR algorithms. These algorithms not only reduce arithmetic complexity but also partially maintain the multiply-accumulate structure, thus resulting in efficient implementations. A set of basic algorithms is derived, together with some rules for combining them. Their efficiency is compared with that of classical schemes in the case of three different criteria, corresponding to various types of implementations. It is shown that this class of algorithms (which includes classical ones as special cases) makes it possible to find the best trade-off corresponding to any criterion.
- Low-area/power parallel FIR digital filter implementation

  This paper presents a novel approach for implementing area-efficient parallel (block) finite impulse response (FIR) filters that require less hardware than traditional block FIR filter implementations. Parallel processing is a powerful technique because it can be used to increase the throughput of a FIR filter or reduce the power consumption of a FIR filter. However, a traditional block filter implementation causes a linear increase in the hardware cost (area) by a factor of L, the block size. In many design situations, this large hardware penalty cannot be tolerated. Therefore, it is important to design parallel FIR filter structures that require less area than traditional block FIR filtering structures. In this paper, we propose a method to design parallel FIR filter structures that require a less-than-linear increase in the hardware cost. A novel adjacent coefficient sharing based sub-structure sharing technique is introduced and used to reduce the hardware cost of parallel FIR filters. A novel coefficient quantization technique, referred to as a scalable maximum absolute difference (MAD) quantization process, is introduced and used to produce quantized filters with good spectrum characteristics. By using a combination of fast FIR

filtering algorithms, a novel coefficient quantization process and area reduction techniques, we show that parallel FIR filters can be implemented with up to a 45% reduction in hardware compared to traditional parallel FIR filters.

- Efficient complexity reduction technique for parallel FIR digital Filter based on Fast FIR algorithm
  The objective of the paper is to reduce the hardware complexity of higher order FIR filter with symmetric coefficients. The aim is to design efficient Fast Finite-Impulse Response (FIR) Algorithms (FFAs) for parallel FIR filter structure with the constraint that the filter tap must be a multiple of 2. In our work we have briefly discussed for L = 4 parallel implementation. The parallel FIR filter structure based on proposed FFA technique has been implemented based on carry save and ripple carry adder for further optimization. The reduction in silicon area complexity is achieved by eliminating the bulky multiplier with an adder namely ripple carry and carry save adder. For example, for a 6-parallel 1024-tap filter, the proposed structure saves 14 multipliers at the expense of 10 adders, whereas for a six-parallel 512-tap filter, the proposed structure saves 108 multipliers at the expense of 10 adders. Overall, the proposed parallel FIR structures can lead to significant hardware savings for symmetric coefficients from the existing FFA parallel FIR filter, especially when the length of the filter is very large.
- Hardware-efficient VLSI implementation for 3-parallel linear-phase FIR digital filter of odd length. Based on fast FIR algorithms (FFA), this paper proposes new 3-parallel finite-impulse response (FIR) filter structures, which are beneficial to symmetric convolutions of odd length in terms of the hardware cost. The proposed 3- parallel FIR structures exploit the inherent nature of the symmetric coefficients of odd length, according to the length of filter, (N mod 3), reducing half the number of multipliers in sub filter section at the expense of additional adders in preprocessing and postprocessing blocks. The overhead from the additional adders in preprocessing and postprocessing blocks stay fixed, not increasing along with the length of the FIR filter, whereas the number of reduced multipliers increases along with the length of the FIR filter. For example, for a 81-tap filter, the proposed A structure saves 26 multipliers at the expense of 5 adders, whereas for a 591-tap filter, the proposed structure saves 196 multipliers at the expense of 5 adders still. Overall, the proposed 3-parallel FIR structures can lead to significant hardware savings for symmetric coefficients of odd length from the existing FFA parallel FIR filter, especially when the length of the filter is large.
- Exploiting coefficient symmetry in conventional polyphase FIR filters.
  The conventional polyphase architecture for linear-phase finite impulse response (FIR) filter loses its coefficient symmetry property due to the inefficient arrangement of the filter coefficients among its sub filters. Although, existing polyphase structures can avail the benefits of coefficient symmetry property, at the cost of versatility and complex sub filters arrangement of the conventional polyphase structure. To address these issues, in this paper, we first present the mathematical expressions for inherent characteristics of the conventional polyphase structure. Thereafter, we use these expressions to develop a generalized mathematical framework which exploits coefficient symmetry by retaining the direct use of conventional FIR filter coefficients. Further, the transfer function expressions for the proposed Type-1/ transposed

Type-1 polyphase structures using coefficient symmetry are derived. The proposed structures can reduce the requirement of multiplier units in polyphase FIR filters by half. We also demonstrate the decimator design using the proposed Type-1 polyphase structure and the interpolator design using the proposed transposed Type-1 polyphase structure. Moreover, the phase and magnitude characteristics of the proposed Type-1/transposed Type-1 polyphase structures are presented. It is revealed via numerical examples that all sub filters of the proposed symmetric polyphase structure possess linear-phase characteristics.

- Efficient FIR filter design using Booth multiplier for VLSI applications.

  The most important criteria for the design and implementation of DSP processor is area optimization and reduction in power consumption. The fundamental block for the design and implementation of the DSP processor is the Finite Impulse Response Filter. The Finite Impulse Response (FIR) Filter consists of three basic modules which are adder blocks, flip flops and multiplier blocks. The performance of the FIR Filter is largely influenced by the multiplier, which is the slowest block out of all. In this paper, the Finite Impulse Response Filter has been proposed using two different multipliers namely Array multiplier and Booth Multiplier and both the proposed FIR filters have been compared for various parameters. The proposed filters are designed using Verilog HDL and is implemented using Xilinx 14.7 ISE tools. An improvement has been obtained both in terms of area and delay. Also, low power consumption and reduction in terms of delay and operational frequency of the booth multiplier makes it highly suitable for the designing of the FIR Filter for low voltage and low power VLSI applications.

This literature survey highlights the key components and existing innovations relevant to designing an Efficient 3-Parallel Polyphase Odd Length FIR Filter utilizing Brent-Kung adders and Booth multipliers. It underscores both the potential and the gaps in current research, paving the way for future work to develop highly optimized hardware architectures for advanced digital filtering applications.

3-REQUIREMENTS

Software

Creating an efficient 3-parallel polyphase odd length FIR filter in Xilinx ISE demands a solid grasp of digital circuit design, hardware description languages (HDLs), and optimization strategies. This type of filter divides the task into three smaller parts, called sub-filters. Each sub-filter runs in parallel, which allows the overall filter to process input signals faster. The goal is to increase data throughput without sacrificing accuracy or quality. To do this, it is vital to understand how polyphase decomposition works. This technique splits the original filter into distinct phases, making it easier and more efficient to process signals. When designing, you need to carefully choose filter coefficients based on the specific frequency response you want. For example, if you aim to pass signals in a certain range of frequencies while blocking others, your filter coefficients must be set precisely. Special attention is needed for filters of odd length since they have a middle coefficient that requires extra handling. This is because the coefficients need to stay symmetric to keep the filter's phase linear — meaning the output signal stays aligned in time. This symmetry ensures the filter doesn't distort the signals passing through it. The architecture itself should be built with

modularity in mind. Breaking the design into smaller, manageable blocks makes it easier to scale or modify later. It also supports easier troubleshooting and maintenance. Efficient resource management is key, meaning the design must use logic elements, memory, and hardware resources wisely to avoid unnecessary complexity. Timing performance must also be prioritized. The filter should operate at the desired clock speed without delay, avoiding issues like timing violations or data loss. Optimizing the pipeline stages and implementing parallel processing help meet strict timing constraints. Overall, this design must balance speed, resource use, and accuracy to produce a reliable, high-performance filter suitable for real-world signal processing tasks.

Development Environment

The development environment utilized for this project is the Xilinx ISE Design Suite, a comprehensive software package tailored for FPGA and CPLD design and development. It is essential to specify the exact version used to ensure reproducibility and compatibility; in this case, the version is 14.7 or later. This version provides a robust set of features necessary for efficient hardware design, simulation, and implementation. The ISE Design Suite is compatible with both Windows and Linux operating systems, allowing flexibility depending on the preferred development environment and available hardware resources.

Within the ISE Design Suite, several core features facilitate the entire design process. The ISE Simulator (ISIM) is employed for simulating and verifying functional correctness of the HDL code before synthesis, enabling early detection of logical errors. The Xilinx Synthesis Tool (XST) converts HDL descriptions into hardware netlists, preparing the design for implementation on target FPGA devices. Additional implementation tools assist in placing and routing the design to meet timing and resource constraints. The suite also includes various simulation tools that enable thorough testing and validation of the design, ensuring that the final hardware implementation performs as intended. Overall, the Xilinx ISE Design Suite provides an integrated environment that streamlines the process from design entry through simulation, synthesis, and implementation, making it an essential tool for FPGA development projects.

Hardware Description Language

HDLs such as VHDL and Verilog play a crucial role in the design and implementation of digital systems, including modules like FIR filters, Brent-Kung adders, and Booth multipliers. These languages enable to model, simulate, and synthesize complex hardware architectures effectively. When selecting between VHDL and Verilog, the decision often depends on the team's expertise and the specific requirements of the project. For instance, VHDL is known for its strong typing and comprehensive syntax, making it suitable for highly detailed and rigorous designs, whereas Verilog offers a more concise syntax that closely resembles traditional hardware description and can be easier for teams with a background in digital logic design. Both languages support the development of efficient, reliable, and scalable modules, which are essential for optimizing performance in digital signal processing applications.

Simulation and Verification

In the development process of complex digital systems, comprehensive simulation and verification play a pivotal role in ensuring that each module functions correctly and reliably within the overall system architecture. This process begins with the

creation of detailed testbenches for each functional block, such as the multiplier, adder, and polyphase filter modules. These testbenches serve as controlled environments where input signals can be systematically applied and outputs can be monitored closely. By simulating a wide array of input scenarios, including typical operating conditions, corner cases, and boundary edge cases, developers can scrutinize the behavior of each module under diverse circumstances. This thorough testing helps uncover potential issues early in the design phase, facilitating necessary adjustments before moving forward with hardware implementation.

The testbenches are meticulously designed to emulate realistic operating environments and to challenge the modules with various signal patterns. For example, input stimuli may include random data streams, fixed amplitude signals, or signals at the extremes of expected ranges. Monitoring the outputs under these conditions allows engineers to verify that the modules produce correct, stable, and predictable results across all tested scenarios. Such detailed validation ensures that the modules not only meet their functional specifications but also behave robustly under diverse conditions, reducing the likelihood of faults or failures once integrated into the complete system.

Beyond simulation within the hardware description environment, the verification process also involves rigorous comparison of the hardware module outputs against established software models or reference algorithms. These models serve as a benchmark for correctness, representing the mathematically accurate or idealized behaviour of each module. By cross-verifying the hardware outputs with these reference models, engineers can confirm that the hardware implementation faithfully replicates the intended logical and mathematical operations. Any discrepancies between the two highlight potential issues in the hardware design or implementation, prompting further investigation and correction.

This dual approach of simulation and cross-verification is essential for early detection and rectification of design errors, significantly reducing the risk of costly revisions later in the development cycle. It ensures that the modules, when finally synthesized into hardware, will perform accurately and reliably as intended. Moreover, this thorough validation process enhances the robustness of the entire system, providing confidence in its correctness and operational stability before proceeding to hardware deployment. Ultimately, such comprehensive verification practices are fundamental to delivering high-quality, dependable digital systems that meet rigorous performance and reliability standards.

4-EFFICIENT 3-PARALLEL POLYPHASE ODD LENGTH FIR FILTER USING BRENT KUNG ADDER AND BOOTH MULTIPLIER

Existing System

The traditional FIR filter architectures predominantly rely on fundamental arithmetic components such as full adders, half adders, carry look-ahead adders (CLA), carry-save adders (CSA), and ripple-carry adders (RCA) to perform the necessary multiply-accumulate (MAC) operations. While these components are essential for basic arithmetic operations, their straightforward implementation often leads to limitations in processing speed and increased hardware resource consumption, especially when dealing with the complex multiplications and accumulations inherent in high-order FIR filters.

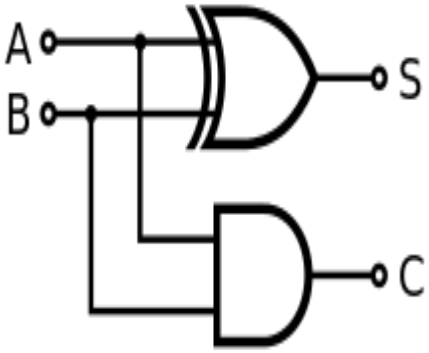
To overcome these challenges, the proposed system integrates a combination of advanced arithmetic techniques, notably the Brent-Kung adder and Booth multiplier. The Brent-Kung adder is renowned for its logarithmic delay and efficient hardware utilization, making it well-suited for high-speed addition operations required during filter computations. Its tree-like structure reduces the logic depth, thereby minimizing propagation delay and improving overall throughput. Similarly, the Booth multiplier accelerates multiplication processes by encoding multiple bits of the multiplier, effectively reducing the number of partial products and thus speeding up the multiplication operation. When combined, these two components significantly enhance the efficiency of the filter's arithmetic operations, enabling faster computation times and reducing power consumption.

Furthermore, the 3-parallel polyphase architecture takes advantage of parallel processing to distribute the computational load across multiple processing elements. This approach not only improves the filter's throughput but also allows for better utilization of hardware resources, leading to a more scalable and flexible implementation. By carefully designing the interconnections and synchronization among these parallel units, the system ensures that the FIR filter can operate at higher clock frequencies with reduced latency. Overall, this advanced design strategy addresses the inherent limitations of conventional FIR filter architectures, paving the way for more efficient, high-performance digital filtering solutions suitable for modern signal processing applications.

Half Adder and Full Adder

A half adder is a fundamental digital circuit used to perform the addition of two single binary digits, commonly referred to as bits. The primary purpose of a half adder is to compute the sum and carry outputs resulting from the addition of these two bits. The two input bits are typically labelled as A and B, where each represents a binary digit that can be either 0 or 1. When these bits are added, the result can be either a simple sum or an overflow that needs to be carried over to the next higher digit in multi-digit binary addition.

The half adder produces two outputs: the sum (S) and the carry (C). The sum output indicates the result of adding the two input bits, while the carry output signifies whether an overflow has occurred, which is especially relevant when dealing with multi-bit binary numbers. In binary addition, the sum can be mathematically expressed as 2 times the carry plus the sum, or mathematically as 2C + S. The simplest implementation of a half adder involves two basic logic gates: an XOR gate and an AND gate. The XOR gate is used to generate the sum (S), because it outputs 1 only when exactly one of the inputs is 1, effectively adding the bits without considering any carry from previous stages. The AND gate produces the carry (C), outputting 1 only when both input bits are 1, indicating an overflow into the next higher bit.



Truth Table			
Input		Output	
A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Fig. 1 Half Adder and it's Truth Table

The Boolean logic expressions for the half adder's outputs are straightforward. The sum (S) can be expressed as A′B + AB′, which is the logical XOR operation, signifying that the sum is 1 when only one of the inputs is 1. The carry (C), on the other hand, is simply AB, which is the logical AND of the two inputs, indicating that a carry occurs only when both bits are 1. To create more complex addition circuits such as a full adder, which can handle carry-in from previous stages, two half adders can be combined along with an OR gate. The first half adder adds the two input bits, while the second half adder adds the carry generated by the first, with the carry-in. An OR gate then combines the carry outputs from both half adders to generate the final carry output for the full adder.

Methodology

For VLSI applications, an efficient 3-parallel polyphase odd-length FIR filter can be implemented using Brent-Kung adder and Booth multiplier. The methodology involves decomposing the odd-length FIR filter into three polyphase components, which are then assigned to three parallel processing units. Each unit utilizes the Brent-Kung adder to perform additions, reducing delay and improving throughput. The Booth multiplier is employed to perform multiplications, minimizing the number of partial products and optimizing resource utilization. The results from each unit are combined to obtain the final filtered output. This approach enables efficient VLSI implementation, offering improved throughput, reduced delay, and optimized resource utilization, making it suitable for high-performance digital signal processing applications.

Brent Kung Adder

The Brent Kung adder is a parallel prefix adder, known for its regular structure and efficient carry propagation mechanism. It is designed to enable high-speed addition of binary numbers by utilizing parallelism in the computation. The adder is suitable for applications where speed is a critical factor, such as in digital signal processing (DSP) circuits.

A key feature of the Brent-Kung adder is its use of simple building blocks called black and gray cells. These cells are small units that do specific jobs in the addition process. Black cells can generate and pass along carry signals, which are necessary to add two bits. Gray cells only generate carry signals but do not pass them along. By arranging these cells carefully, the Brent-Kung adder can quickly find the total sum of two large binary numbers, even if they are many bits long

To understand why this matters, think about a computer that needs to do millions of calculations

.

5-RESULT

ISSN: 2456-4265
IJMEC 2025

per second. Fast adders like the Brent-Kung help improve speed and efficiency. This type of adder is especially useful in high-performance computers, digital signal processors, and other devices that need quick calculations.

In practice, the Brent-Kung adder works by first breaking down the addition into smaller parts using the black and gray cells. It then combines the partial results to produce the final sum. This process reduces the time needed to handle large numbers compared to traditional adders. Because it can add large numbers quickly, the Brent-Kung adder is a favorite choice for designing efficient digital circuits.

Black Cells:

Black cells represent the carry-save adders within the design of a Brent-Kung adder. These cells are key parts of the circuit and handle the main calculations for addition. Each black cell receives two input bits, which are labeled as a and b. These bits can come from either previous calculations or from other parts of the circuit as it processes the addition. The purpose of each black cell is to take these two bits and generate two outputs, called s and c. The s output represents the sum of the two input bits, similar to how adding two numbers works in basic arithmetic. The c output, on the other hand, is the carry result that might be carried over to the next stage of addition.

- **Gray Cells:**

Gray cells in the Brent-Kung adder are used to show the carry-propagate operations. These cells play a key role in how the adder handles addition. Each gray cell takes in two bits, named s and c. The s bit is one of the bits being added, and c is the carry bit coming from a previous addition step. The gray cell then processes these inputs to generate a single output, called the sum. This sum output is the final result of adding those two input bits.

This design helps speed up the addition process, especially when dealing with large numbers. Instead of waiting for the carry to ripple through every previous bit, the gray cells handle some parts in parallel. For example, if you add two binary digits like 1 and 1, the sum is 0 with a carry of 1. The gray cell uses its inputs (s and c) to determine this sum

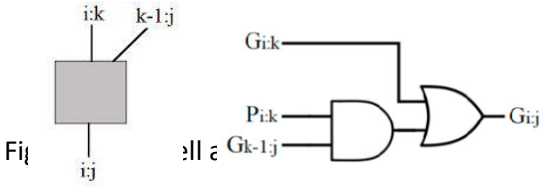
For example, if a is 1 and b is 0, then the s output would be 1, since 1 + 0 equals 1. The c output would be 0 because there's no carry in this case. If both a and b are 1, then the s output becomes 0 (since 1 + 1 equals 10 in binary), and the c output becomes 1, which would then be passed on as a carry to the next grouping of bits. These simple operations happen simultaneously within each black cell, allowing the entire adder to process bits in parallel and speed up the calculation.

Understanding the black cells is important because they form the core logic of the carry-save addition process. Carry-save adders are used to efficiently add multiple binary numbers by avoiding the delay caused by carry propagation that normally happens in standard adders. Instead of waiting for carries to be passed through each digit, these cells handle carry generation locally at each stage. This makes the overall addition process much faster, especially when dealing with large numbers or multiple sums.

In the context of the Brent-Kung adder, these black cells are laid out in a specific pattern to optimize speed and reduce the size of the circuit. Their placement ensures that carry signals are propagated quickly through the system without unnecessary delays. This architecture allows for faster addition times, which is especially useful in high-speed computing and digital signal processing applications

.

quickly. If s is 1 and c is 0, it produces a sum of 1. If both are 1, it produces a sum of 0 and carries over 1. Understanding these gray cells helps explain why the Brent-Kung adder can perform faster than simple ripple-carry adders. They split the computation into several parts that work at the same time. This structure reduces delay and makes addition more efficient. These cells are essential building blocks that enable the entire adder to handle larger numbers quickly. The design requires careful arrangement so that the sum and carry bits flow correctly through the system, ensuring accurate and fast results.



- **Buffer Cells:**

A buffer functions by directly passing the input signal to the output without altering its value, effectively serving as an intermediary that ensures signal integrity and stability. By doing so, it helps to prevent signal degradation that can occur over long distances or through complex circuitry. Additionally, buffers play a crucial role in minimizing the load on the critical path of a circuit, as they isolate stages, reduce the fan-out, and help maintain the desired timing characteristics. This ensures that the overall system operates efficiently and reliably, with roved

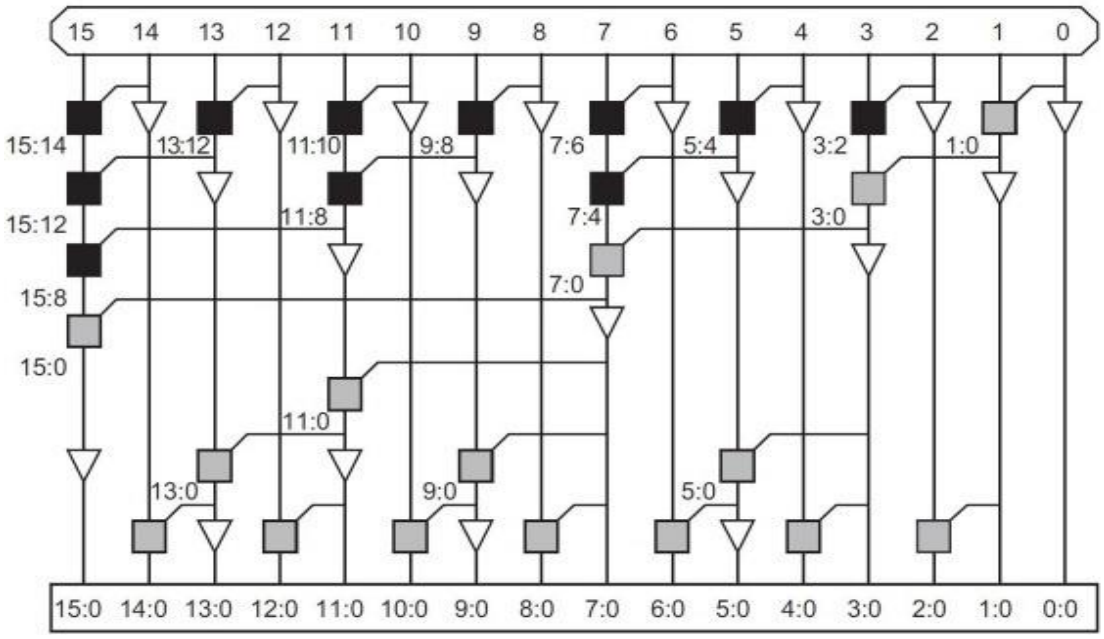
ignal quality and reduced delay, making buffers essential components in digital and analog circuit design. Fig. 6.4 Brent Kung Adder (4 bit)

Fig. 6.5 Brent Kung Adder (16 bit)

The adder takes two binary numbers, A and B, of equal length.For each bit position, generate the generate (G) and propagate (P) signals:

Generate (G): Indicates if a carry will be generated at this position regardless of the incoming carry.

$G_i=A_i \wedge B_i$



Propagate (P): Indicates if a carry-in will propagate through this position.

$P_i=A_i \vee B_i$

ISSN: 2456-4265
IJMEC 2025

The Brent-Kung adder uses a prefix computation tree to efficiently combine generate and propagate signals. The tree performs "group generate" and "group propagate" calculations in a hierarchical manner.

The structure is divided into logarithmic levels:

Up-sweep (reduce) phase: Combines signals from the least significant bits upward, computing group generate and propagate signals for larger groups.

Down-sweep phase: Uses these group signals to determine the carry-in for each bit.

Carry Calculation: Using the prefix signals, the carry for each bit is computed

$C_{i+1}=G_i \vee (P_i \wedge C_i)$

Sum Computation: The sum bits are then calculated using the carry-in:

$S_i=A_i \oplus B_i \oplus C_i$

Working of Booth Multiplier

The Booth multiplier is a hardware algorithm used for multiplying binary numbers efficiently, especially in computer architecture and digital systems. Its primary advantage is reducing the number of addition and subtraction operations needed, which leads to faster multiplication processes. The Booth algorithm works by encoding the multiplier in a way that minimizes the number of necessary operations, particularly handling sequences of consecutive ones more efficiently.

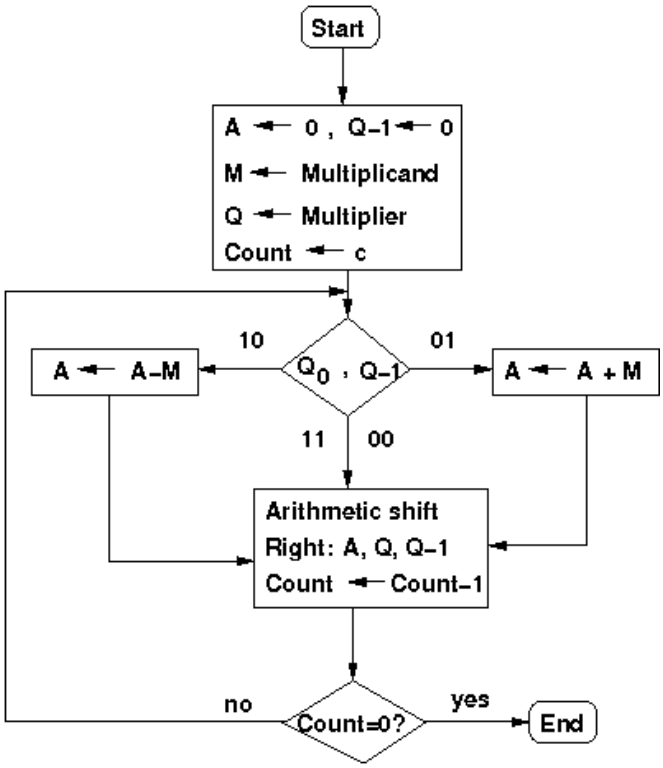


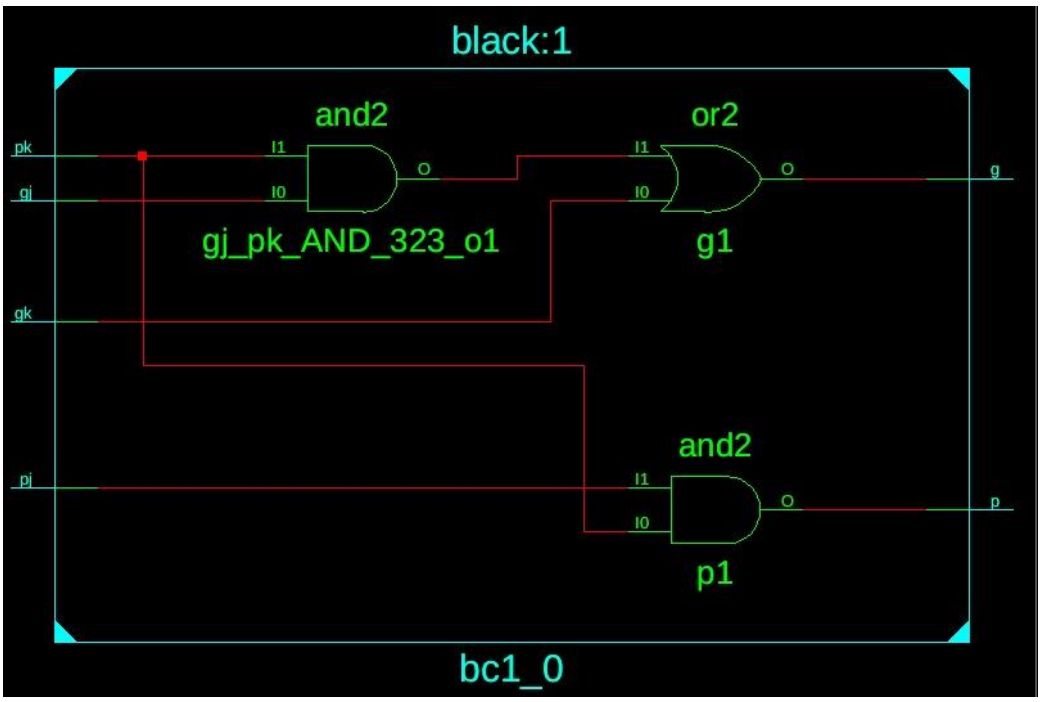
Fig. Booth Multiplier Algorithm Flowchart

The process begins by examining the bits of the multiplier, typically starting from the LSB. The algorithm looks at pairs of bits — the current bit and the previous bit — to determine what operation to perform. If the pair is 01, it indicates that the multiplicand should be added; if it is 10, the multiplicand should be subtracted; and if it is 00 or 11, no operation is required for that step. This encoding is known as Booth's encoding.

During each step, based on the bits being analyzed, the multiplicand is either added, subtracted, or ignored, and the result is shifted accordingly. The

shifting aligns the partial products correctly for the next iteration. By doing this, the Booth multiplier effectively compresses runs of ones and zeros, reducing the total number of addition/subtraction operations compared to straightforward multiplication algorithms.

Overall, the Booth multiplier's working mechanism involves analyzing the bits of the multiplier, encoding these bits to minimize operations, and performing a sequence of additions, subtractions, and shifts. This approach accelerates multiplication, especially for large binary numbers, making it a widely used technique in digital computing systems.In a practical implementation of this design, various modules come into play. The black, gray, and buffer modules shown in the RTL symbols and schematics work together to facilitate the adder's operation. The black modules represent the carry-save operations, while the gray modules represent the carry-propogate operations. Buffer modules are used to strengthen signals and prevent delays caused by weak or noisy signals. All these modules work together seamlessly to implement the Brent-Kung adder efficiently.



6-CONCLUSION

In conclusion, the project aimed at achieving a high-speed VLSI implementation of an odd-length FIR filter has been successfully completed. Leveraging the strengths of the Brent Kung adder and Booth multiplier, we addressed the critical need for efficient and speedy FIR filtering in digital signal processing applications. The use of the Brent Kung adder contributed significantly to the speed enhancement of the FIR filter. The parallelism in its structure facilitated faster addition, crucial for real-time signal processing. The design's efficient carry lookahead mechanism minimized delays associated with carry propagation, a key factor in achieving high-speed computation. The inclusion of the Booth multiplier complemented the Brent Kung adder by optimizing the multiplication stage of the FIR filter. The Booth multiplier's ability to reduce the number of partial product additions positively impacted overall filter performance.

REFERENCES

[1]. K. K. Parhi, VLSI Digital Signal Processing System: Design and Implementation (Wiley, New York, 1999).

[2]. L. K. Phimu and M. Kumar, "Design and implementation of area efficient 2-parallel filters on FPGA using image system" in proc. ICECDS, 2017.

[3]. Z.-J. Mou, and P. Duhamel, "Short-length FIR filters and their use in fast nonrecursive filtering" IEEE Trans. Signal Process., vol. 39, no. 6, 1991.

[4]. D. A. Parker, and K. K. Parhi, "Low-area/power parallel FIR digital filter implementation," J. VLSI Signal Process. Syst, vol. 17, 1997.

[5]. J. Selvakumar and Vidhyacharan Bhaskar, "Efficient complexity reduction technique for parallel FIR digital Filter based on Fast FIR algorithm," International Journal of Computer Applications, Vol. 55, 2012.

[6]. Y.C. Tsao, and K. Choi,, "Hardware-efficient VLSI implementation for 3-parallel linear-phase FIR digital filter of odd length" in proc. IEEE ISCAS, 2012.

[7]. Y.C. Tsao, and K. Choi,, "Hardware-efficient VLSI implementation for 3-parallel linear-phase FIR digital filter of odd length" in proc. IEEE ISCAS, 2012.

[8]. Q. Tian, Y. Wang, G. Liu, X. Liu, J. Diao, and Hui Xu "Hardware-efficient parallel FIR filter structure based on modified Cook-Toom algorithm" in proc. IEEE APCCAS, 2018.

[9]. K Anjali Rao, Abhishek Kumar, Neetesh Purohit, "Efficient implementation for 3-parallel linear-phase FIR digital odd length filters" in proc. IEEE CICT, 2020.

[10]. A.Kumar, S. Yadav and N. Purohit, "Exploiting coefficient symmetry in conventional polyphase FIR filters," IEEE Access, vol.7, 2019.

[11]. S.Y. Park, and Pramod K. Meher, "Efficient FPGA and ASIC realizations of DA-based reconfigurable FIR digital filter," IEEE Trans. Circuit and Syst.II, vol.61, no. 7, 2014.

[12]. T. Vamshi Krishna, Niveditha S, Mamatha G. N, Sunil M. P. "Simulation study of brent Kung adder using cadence tool," International Journal of Advanced Research, Ideas, and Innovations in Technology, 2018.

[13]. D. K. Kahar and H. Mehta, "High-speed Vedic multiplier used Vedic mathematics" in Proc. ICICCS, 2018.

[14]. Rajesh K., Reddy G. "FPGA implementation of multiplier accumulator unit using Vedic multiplier and reversible gates" in proc. ICISC, 2019.

[15]. S. Nagaria, A. Singh, and V. Niranjan "Efficient FIR filter design using Booth multiplier for VLSI applications" in proc. IEEE CPCT (GUCON), 2018 Authorized