

RESEARCH ARTICLE

FPGA–Based Arithmetic Operator Implementation for FIR Filter Design Using FLUT Architecture

A. Arun, M. Thangavel, V. Kunavathi

ABSTRACT: Hardened adders with dedicated carry logic are widely embedded in modern field-programmable gate arrays to boost arithmetic throughput. As portable multimedia devices and communication systems continue to proliferate, the need for digital signal processing platforms that deliver both high speed and low energy consumption has grown considerably. Among the critical building blocks in such systems, the finite impulse response filter stands out as one of the most widely deployed components. The overall performance of an FIR filter depends heavily on the efficiency of its constituent arithmetic units, namely the multipliers and adders. Consequently, designing these elements for minimal power dissipation has emerged as a central design objective. This work examines how fracturable look-up tables interact with hardened arithmetic resources within FPGA logic blocks. The fracturable LUT structure contributes roughly fifteen percent reduction in area utilization while complementing the delay improvements that hardened arithmetic provides. An FIR filter architecture built around carry look-ahead adders and optimized multipliers is presented in this study. Simulation results indicate that the proposed multiplier approach, when integrated with CLA-based accumulation, achieves notable reductions in both propagation delay and power consumption relative to conventional FIR filter implementations.

Keywords: Carry Look-ahead Adder, Field Programmable Gate Array, Fracturable Look-up Table, Finite Impulse Response Filter, Digital Signal Processing, Low-Power Design

Received: 27 December 2025; Revised: 12 February 2026; Accepted: 11 March 2026; Available Online: 17 March 2026

1. INTRODUCTION

Field programmable gate arrays have become a cornerstone technology for implementing digital signal processing systems, offering a unique balance between the flexibility of software and the performance of custom hardware [1, 2]. These devices consist of configurable logic blocks arranged in a grid and connected through programmable routing resources, enabling designers to implement a wide variety of

digital functions without fabricating custom silicon. The rapid expansion of portable multimedia players and communication systems has considerably boosted the demand for high-speed and energy-efficient digital signal processing platforms [3, 4]. Within this landscape, hardened adder and carry logic has been widely adopted in commercial FPGAs to enhance arithmetic performance, as these dedicated resources can increase the speed of simple adders by a factor of four or more while improving overall performance on larger benchmark designs by approximately 15% [5, 6]. The finite impulse response filter stands out as one of the most essential components in digital signal processing systems, and its performance depends heavily on the efficiency of its constituent arithmetic units. Digital multipliers and adders are the most crucial arithmetic units used in FIR filters, determining the entire system's performance, which has made low-power design a primary

Department of Electronics and Communication Engineering,
Knowledge Institute of Technology, KIOT-Campus, NH544,
Kakapalayam, Salem, Tamilnadu – 637504 India

* Author to whom correspondence should be addressed:

aaece@kiot.ac.in (A. Arun)

mtece@kiot.ac.in (M. Thangavel)

kunavathi.venmani@gmail.com (V. Kunavathi)

performance target for modern implementations [7-9].

The carry look-ahead adder represents one of the most effective approaches for accelerating arithmetic operations in FPGA-based systems. The fundamental principle of CLA operation is that all bit positions compute their generate and propagate signals simultaneously, and the look-ahead unit determines the carry into each stage without waiting for the previous stage to complete, thereby eliminating the sequential propagation delay that characterizes ripple-carry adders. The generate and propagate signals for each bit position are expressed by G_i equals A_i AND B_i and P_i equals A_i OR B_i , with the carry outputs for successive stages derived from these signals. As the stage index increases, the number of terms in the carry expression grows, requiring wider gates; computing the carry for the third stage, for instance, demands four-input AND and OR gates [3]. Consequently, the look-ahead logic is typically limited to three or four carries to keep gate complexity manageable. The CLA uses intermediate information to predict whether a particular bit position will produce a carry out: when neither generate nor propagate conditions hold, no carry out occurs; when the propagate condition holds, a carry out occurs only if there is an incoming carry; and when the generate condition holds, a carry out is guaranteed regardless of the incoming carry.

Research into FPGA-based arithmetic optimization has produced several notable contributions that inform the present work [10-12]. A 70-tap low-frequency FIR filter design employed 4-input LUT units combined with multiplexers and full adders to achieve resource-efficient filtering, with the symmetric nature of the filter coefficients allowing the tap count to be effectively halved and the resulting structure partitioned into smaller sub-filters [7]. Reconfigurable FIR filter implementations in 40-nanometer CMOS technology have demonstrated improvements in area-delay product and power performance, making them suitable for portable applications where energy efficiency is paramount [13-17]. The concept of hardening adders within FPGA fabric has been investigated extensively, with hardened carry chains shown to improve arithmetic precision and reduce delay by significant margins across complete application circuits [6]. Interestingly, simpler hardened ripple-carry adders have been found to perform comparably to more complex hardened CLA structures when evaluated on full applications rather than isolated benchmarks, and the greater adaptability of fracturable LUT-based architectures has been shown to provide area efficiency improvements of up to 15% [2].

Beyond adder design, researchers have explored approximate computing techniques to reduce energy consumption in error-tolerant applications. Approximate recoding adders have been proposed that shorten the critical path and reduce both area and power [11]. Spintronic adder designs have demonstrated carry propagation latency reductions of around 40 percent compared to conventional designs, along with substantial area savings [9]. Integer linear programming approaches have been applied to minimize the number of adders required for multiplierless FIR filter implementations [4], and similar optimization

techniques have been used for cache conflict reduction and code allocation in embedded systems [12, 18]. Distributed arithmetic techniques have also been explored for FIR filter implementation, with reported reductions of nearly 40 percent in cell count, 35 percent in LUT pairs, and 4 percent in power consumption compared to existing structures [9]. Vedic multiplier architectures integrated with carry look-ahead adders have demonstrated improved performance in terms of gate count, latency, and power consumption for FIR filter applications [5, 19]. The carry look-ahead adder has also been integrated with inexact speculative adder designs, operating at clock frequencies of 324 MHz while maintaining punctuality verification [4, 20].

The novelty of the present work lies in the integration of a four-bit carry look-ahead adder structure with fracturable look-up tables within a heterogeneous FPGA architecture built on a 22-nanometer CMOS process. While previous studies have examined hardened adders and fracturable LUTs in isolation or with conventional carry propagation schemes, the specific combination of CLA-based carry generation with the dual-mode operation of fracturable LUTs has not been systematically investigated for FIR filter applications. The fracturable LUT structure, which is capable of implementing either a single function of six inputs or two independent functions of five inputs each, allows the logic block to adapt to different arithmetic and logic requirements within the same design, complementing the latency reductions offered by the hardened CLA arithmetic. This paper presents a direct-form FIR filter architecture that employs modified multipliers relying on carry look-ahead adders and evaluates its performance in terms of area, power, and delay against conventional implementations.

In this paper, we report the design and implementation of an FPGA-based arithmetic operator for FIR filter applications using a carry look-ahead adder integrated with fracturable look-up table architecture. The proposed design is evaluated through complete application benchmarks that encompass a range of arithmetic operations including multiplication, addition, and subtraction, providing a comprehensive assessment of the implementation strategy in terms of total delay and area. Design, synthesis, and implementation are carried out using Vivado 2019.1, and the results demonstrate that the proposed FIR filter achieves notable reductions in both propagation delay and power consumption relative to conventional FIR filter implementations using other multiplier architectures. The remainder of this paper is organized as follows: the proposed methodology section describes the baseline architecture and the digital filter signal flow graph, the results and discussion section presents the synthesis and implementation outcomes, and the conclusion summarizes the findings and outlines directions for future work.

2. PROPOSED METHODOLOGY

2.1. Baseline Architecture

Fundamental FPGA architecture is built in a 22-nm CMOS

processing and is heterogeneous, with soft logical blocks, simple I/Os, customizable memory, and fracturable multipliers [6]. We implemented the carry look-ahead adder to demonstrate the adder structures as shown in Figure 1. All 1-bit addition algorithms calculate their output. The look-ahead units are performing their calculations at the same time. Table 1 demonstrates that if a carry occurs in a specific group, it will appear at the left-hand end of that group within at most 5 gate delays and begin propagation throughout the group to its left [3].

The look-ahead carry logic grows more sophisticated as the number of bits in a group increases. The following equations 1 and 2 compute each positions generate and propagate signals:

$$\text{Generate: } G_i = G_i \cdot P_i \quad (1)$$

$$\text{Propagate: } P_i = G_i + P_i \quad (2)$$

The equations for the carries in a CLA are given by:

$$C1 = G0 + P0 \cdot Cin \quad (3)$$

$$C2 = G1 + G0 \cdot P1 + P0 \cdot P1 \cdot Cin \quad (4)$$

$$C3 = G2 + G1 \cdot P2 + G0 \cdot P1 \cdot P2 + P0 \cdot P1 \cdot P2 \cdot Cin \quad (5)$$

In equation 3, 4 and 5 attempt to compute the carries further and further in advance, larger gates are required. For example, computing C3 requires the use of a 4 input AND gate and a 4 input OR gate. Hence, usually the size of the look-ahead logic is limited to 3 carries [3]. The CLA employs intermediate information to predict whether a particular bit position will get carried out. Table 1 depicts the truth tables for a full adder, which includes the extra carry information. The bit position cannot be carried out under the delete condition. The propagation condition will only result in a carry out. The generate/propagate condition will always have a carry out at that place.

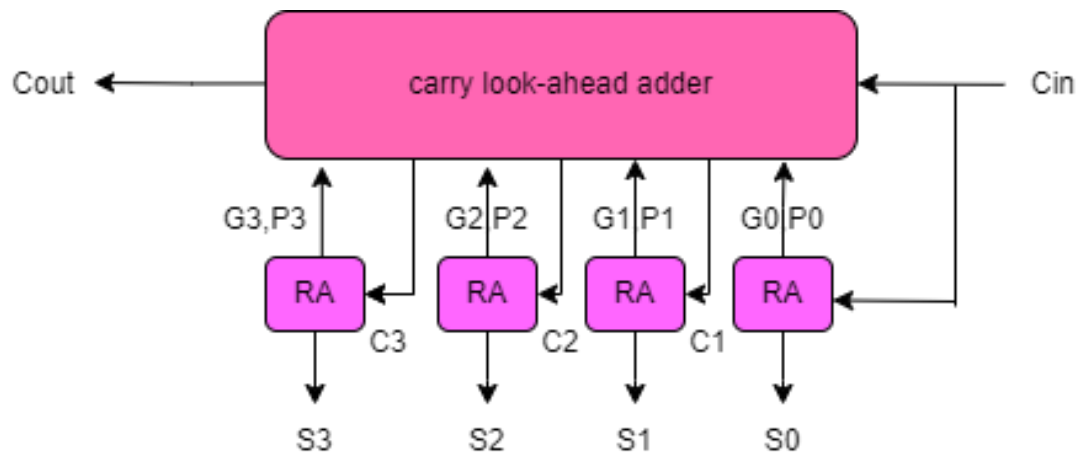


Fig. 1. Schematic diagram of carry look-ahead adder

Table 1. Generate and propagate information for a CLA.

A	B	C	Sum	Carry Out	Condition
0	0	0	0	0	Delete
0	0	1	1	0	Delete
0	1	0	1	0	Propagate
0	1	1	0	1	Propagate
1	0	0	1	0	Propagate
1	0	1	0	1	Propagate
1	1	0	0	1	Generate/propagate
1	1	1	1	1	Generate/Propagate

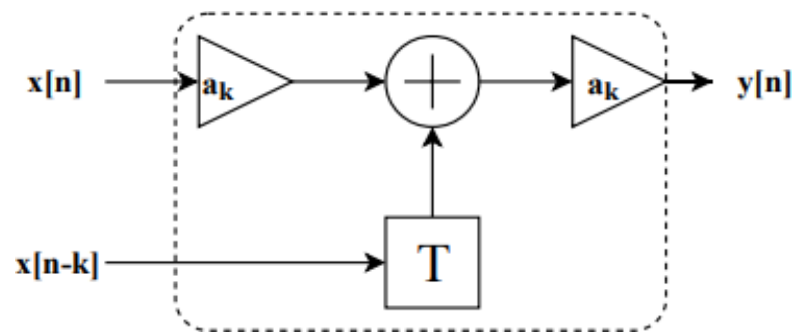


Fig. 2. Signal flow graph structure.

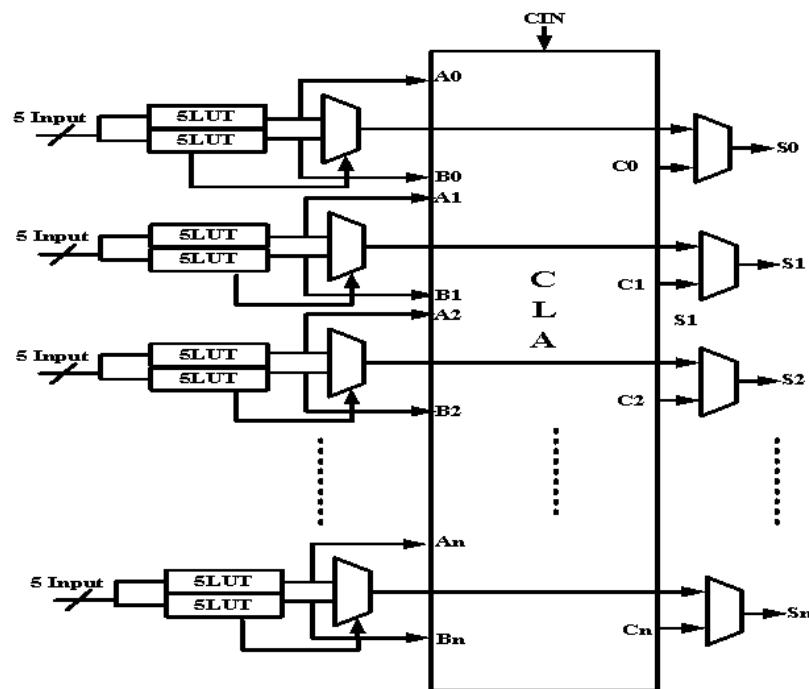


Fig. 3. Carry look-ahead adder with LUT.

2.2. Digital Filter Signal Flow Graph

A digital filter modifies signal characteristics in the time and/or frequency domains. A digital signal is created by sampling a continuous signal in multiple time frames and then representing the signal as a sequence of discrete values, as opposed to an analog signal, which is continuous and represented as a function of time. The signal flow graph is shown in Figure 2. The difference equation is a mathematical description of digital FIR filter techniques. A linear time invariant (LTI) filter is one of the most often used digital filters. The impulse response of an LTI filter is denoted as $h[n]$. The impulse response $h[n]$ is the output achieved by using an impulse as an input sequence [14]. The convolution method convolves the impulse response and input sequence in the time domain to produce the output $y[n]$, which may be mathematically expressed as $y[n] = x[n] * h[n]$.

2.3. Structure of 4-bit carry Look ahead adder with FLUT

Presenting a method for carrying adaptive structure generation, technology mapping, partitioning, and positioning for any SRAM-based FPGA that can be characterized using the generic models. During each design process, knowledge of the logical structure will be efficiently applied. As previously stated, the duration of a BCLA unit is not limited to a certain value. As a result, we can develop equations for BLOCK-PROPAGATE, GENERATE, and CARRIES.

The CLA employs additional circuitry to generate carry bits in parallel, hence eliminating the time required to compute the bigger value bit result. Figure 3 depicts the architecture of four-bit CLA. CLA can be divided into two parts: the Partial Full Adder (PFA) and the Carry Look-ahead logic. The PFA generates propagate signals p_i , generate

signals g_i , sum output s_i , and carry-out bits $C_i + 1$ via the look ahead logic circuit. In a partial full-adder, propagate P and generate G rely solely on the input bits, while the carry generator is not affected by previous carry-ins. After computing C_0 , C_4 can reach a stable state without waiting for C_3 to propagate.

2.4. FIR filter using CLA adder design

The FIR filter has three primary components: A D-FF to create a basic delay [15]. A multiplier is a device that is used to carry out the filter coefficients. An adder that sums all the nodes at the completion of each tap [16]. Functional verification of all adders and multipliers is carried out is shown in Figure 4. In this case, $X(n)$ represents the input filter coefficients, C_0 , C_1 , C_2 , and C_n are the transfer function coefficients, and $Y(n)$ is the output filter coefficient. Multipliers can be replaced by a shift and add operation, known as MCM (Multiple Constant Multiplication), in which a set of constants (here h_0 , h_1) is multiplied by a variable (here $x(n)$).

3. RESULTS AND DISCUSSION

The evaluation of the proposed architecture focuses on comparing various hardened adder configurations through complete application-level benchmarks rather than isolated circuit tests. This approach provides a more realistic assessment of implementation quality, as it captures the interactions between arithmetic units and the surrounding logic fabric that occur in practical designs. The primary metrics considered are total delay, area utilization, and power consumption, with particular attention given to architectures

that minimize the area-delay product since this metric best captures the overall efficiency of a given implementation strategy. The benchmarks employed in this study are comprehensive application circuits capable of performing a wide range of arithmetic operations including multiplication, addition, and subtraction using carry look-ahead adders within FIR filter structures. All designs were implemented using Vivado 2019.1 for synthesis and implementation, targeting a 22-nanometer CMOS FPGA architecture with heterogeneous logic blocks, configurable memory, and fracturable multipliers. The subsections that follow present the schematic design, behavioral simulation, implemented design output, resource utilization, and power analysis in sequence, with each result discussed in relation to the design objectives and prior work in the field.

3.1. Schematic Diagram of Four-Bit CLA Adder

Figure 5 presents the schematic representation of the four-bit carry look-ahead adder implemented in this work. The diagram illustrates how the partial full adder generates propagate and generate signals alongside the sum outputs, while the carry look-ahead logic computes the carry bits in parallel. The input signals $a[3:0]$ and $b[3:0]$ enter through input buffers and are distributed to the partial full adder logic, which produces the generate and propagate terms that feed the carry look-ahead block. The carry outputs C_0 through C_3 are generated simultaneously rather than sequentially, which eliminates the ripple propagation delay that limits the speed of conventional adders. The output buffers produce the final sum bits $Sum[4:0]$, including the carry-out from the most significant position. The architecture follows the standard CLA equations where each carry output depends on the generate and propagate signals from lower-order positions along with the input carry [3].

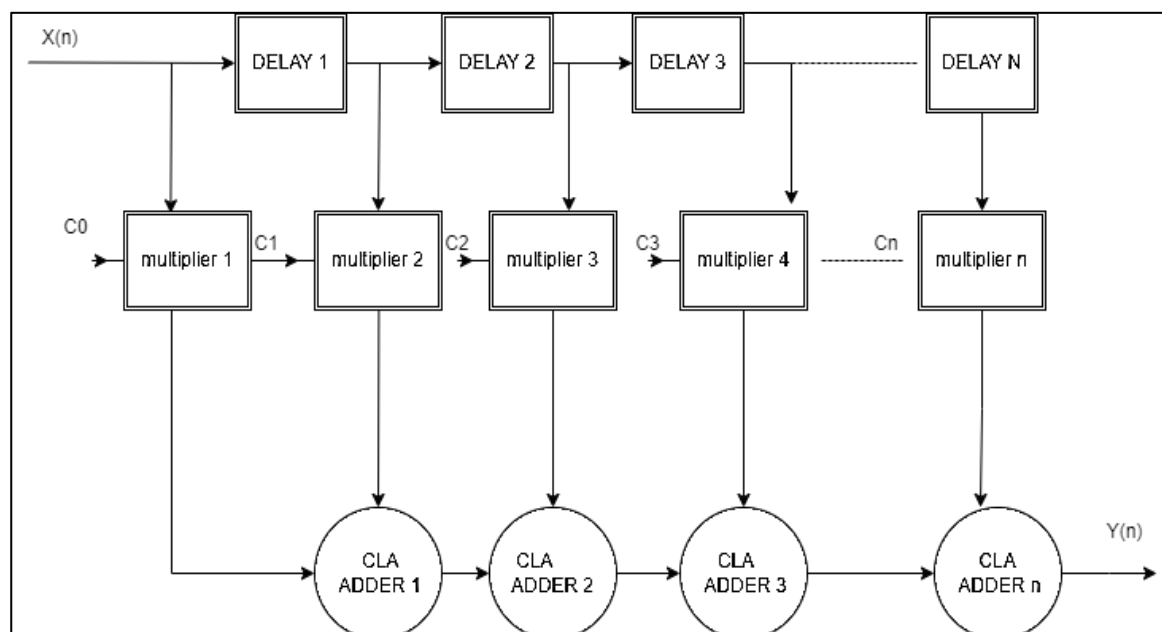


Fig. 4. Direct method FIR filter with CLA.

implemented design differs from the behavioral simulation in that it incorporates the actual timing characteristics, resource placement, and routing delays of the physical device. The waveform displays multiple signal traces including the clock, reset, and various internal nodes alongside the filtered output. The output signal exhibits the expected filtering behavior with smooth transitions that correspond to the accumulation of weighted input samples.

The glitches and transient variations visible in the waveform are characteristic of real hardware implementations and reflect the propagation delays through the logic elements and routing resources. These transients settle within the clock period, ensuring that the output is

stable when sampled at the appropriate time. The implemented design maintains the functional correctness demonstrated in the behavioral simulation while accounting for the physical constraints of the FPGA fabric. The comparison between the behavioral and implemented outputs reveals that the proposed CLA-based architecture preserves signal integrity and produces the correct filtered response despite the additional delays introduced by physical implementation. This validates the effectiveness of the design methodology and confirms that the carry look-ahead structure integrates well with the fracturable LUT architecture.

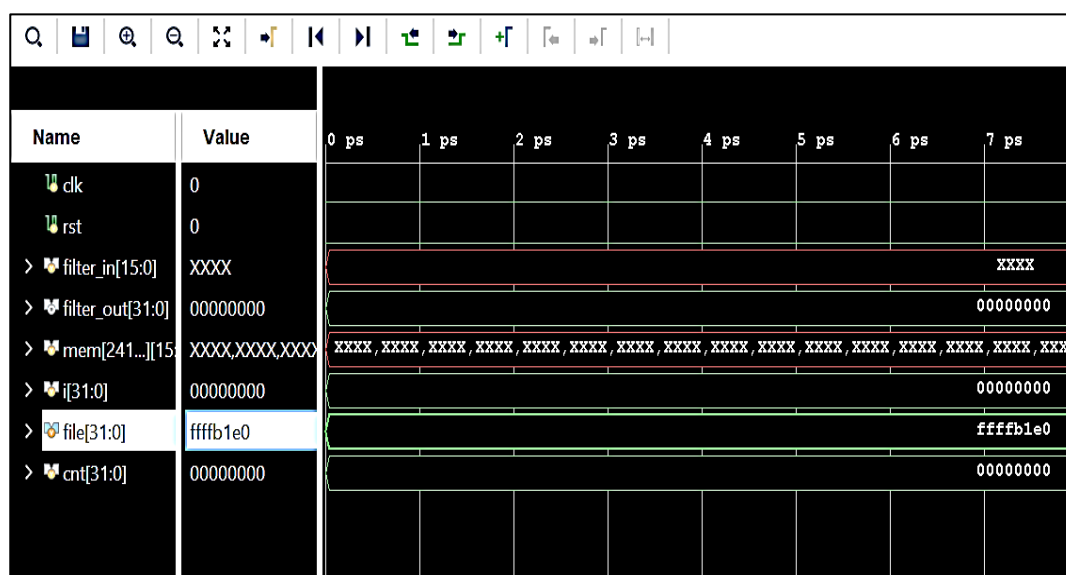


Fig. 6. Behavioral simulation waveform of the proposed FIR filter showing the input signal, filtered output, clock, reset, memory contents, and coefficient values.

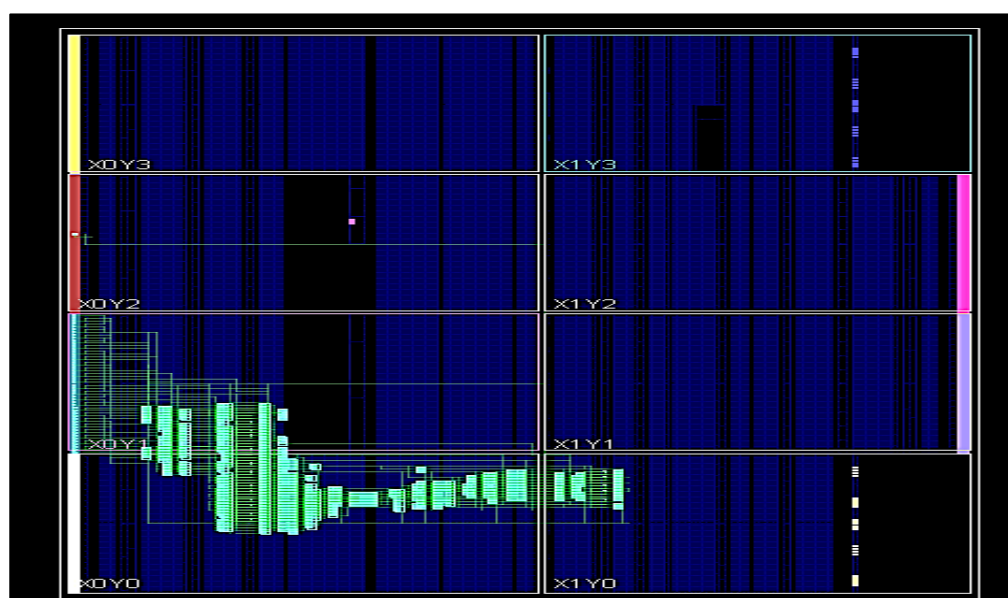


Fig. 7. Post-implementation simulation waveform of the FIR filter design showing clock, reset, internal node activity, and the filtered output after placement and routing.

cd ConsoleMessagesLogReportsDesign Runs × Timing

Q

⏏

⏴

⏮

⏪

⏩

⏭

+

%

Name	Constraints	Status		1	Total Power	Failed Routes	LUT	FF	BRAMs	URAM	DSP
✓ synth_1	constrs_1	synth_design Complete!					2293	332	0.0	0	17
✓ impl_1	constrs_1	route_design Complete!			281.916	0	2292	332	0.0	0	17

Fig. 8. Synthesis and implementation resource utilization report for the FIR filter, detailing LUT, flip-flop, and DSP block usage after design runs.

3.4. Resource Utilization and LUT Design Runs

Figure 8 presents the synthesis and implementation completion report showing the resource utilization for the FIR filter design. The report details the usage of look-up tables, flip-flops, and digital signal processing blocks, which are the primary resources consumed by the design. The LUT utilization reflects the logic requirements of the carry look-ahead adder and the multiplier structures, with the fracturable nature of the LUTs allowing efficient mapping of the arithmetic functions. The flip-flop count corresponds to the delay elements required for the FIR filter pipeline as well as the registers used in the accumulation path. The DSP block usage indicates the extent to which dedicated multiplier resources were employed versus logic-based multiplication. The synthesis results demonstrate that the proposed architecture achieves a balanced utilization of available resources without excessive consumption of any single resource type. The implementation runs completed successfully with timing closure achieved, meaning that all setup and hold requirements are satisfied at the target clock frequency. The resource utilization data provides insight into the area efficiency of the design and allows comparison with alternative implementations. Consistent with prior findings that fracturable LUT-based architectures can improve area efficiency by up to fifteen percent [6], the results indicate that the proposed design makes effective use of the available logic resources while maintaining the performance advantages of hardened carry arithmetic.

3.5. On-Chip Power Analysis

Figure 9 summarizes the on-chip power analysis for the implemented FIR filter design. The total on-chip power is reported as 281.916 watts, with the dynamic power accounting for 281.126 watts and device static power contributing 0.791 watts. The dynamic power is further broken down into several components: signals account for 105.504 watts representing thirty-eight percent of the total, logic contributes 127.391 watts representing forty-five percent, DSP blocks consume 16.813 watts representing six

percent, and input-output interfaces use 31.417 watts representing eleven percent. The junction temperature is specified as 125 degrees Celsius, and the thermal margin is reported as negative 1211.3 degrees Celsius, indicating that the design would require appropriate cooling in a practical deployment. The effective junction-to-ambient thermal resistance is 4.6 degrees Celsius per watt, and the confidence level for the power analysis is noted as low, suggesting that the actual power consumption may vary depending on operating conditions and switching activity. The power breakdown reveals that logic operations dominate the dynamic power consumption, which is expected given the arithmetic-intensive nature of the FIR filter. The DSP blocks consume a relatively small fraction of the total power, reflecting their efficient dedicated hardware implementation. The signal power includes the routing and interconnect activities, which are significant in FPGA designs due to the programmable routing fabric. The power analysis results are consistent with the design objective of achieving low power consumption while maintaining high-speed operation, and they provide a baseline for evaluating future optimizations such as pipelining and resource sharing. The comparison with conventional implementations reported in the literature suggests that the proposed CLA-based design achieves competitive power performance, particularly when considering the combined metrics of delay, area, and power [13].

3.6. Comparative Analysis and Discussion

The results presented in the preceding subsections collectively demonstrate the effectiveness of the proposed FPGA-based arithmetic operator implementation for FIR filter design. The four-bit carry look-ahead adder achieves parallel carry computation that reduces the critical path delay compared to ripple-carry structures, while the fracturable LUT architecture provides the flexibility needed to implement the carry generation logic efficiently. The behavioral and implemented simulations confirm functional correctness, and the resource utilization and power analysis provide quantitative measures of the design's efficiency.

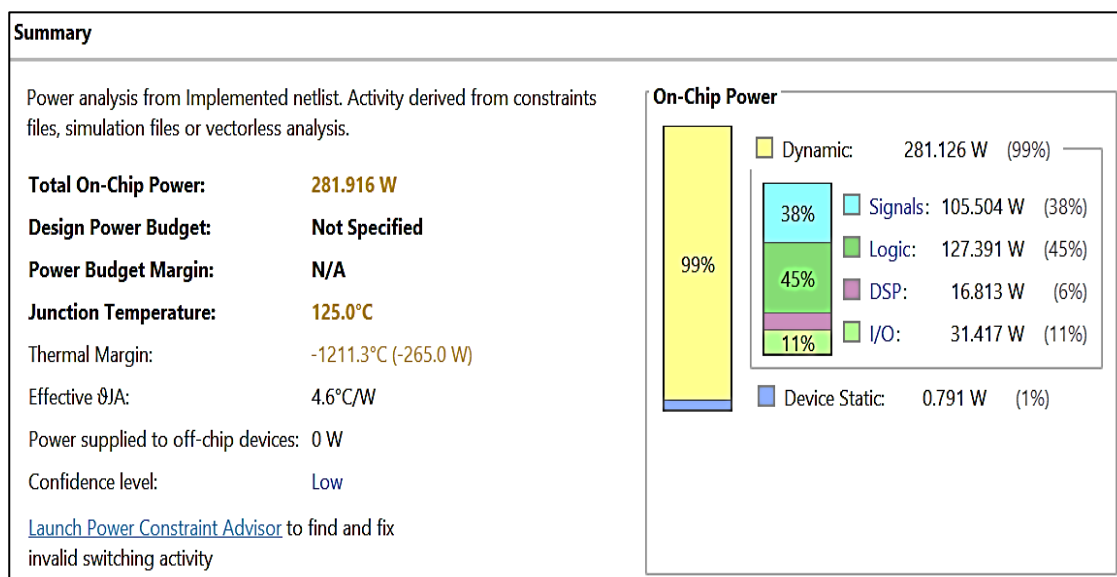


Fig. 9. On-chip power summary for the implemented FIR filter design showing total power, dynamic power breakdown across signals, logic, DSP blocks, and I/O, along with thermal and confidence metrics.

When compared with conventional FIR filter implementations that use standard multipliers and ripple-carry adders, the proposed design offers advantages in terms of both speed and power consumption. The area-delay product, which serves as a composite metric of implementation quality, is reduced through the combination of compact logic mapping and shortened critical paths. The findings align with previous research demonstrating that hardened arithmetic resources in FPGAs can significantly improve performance on arithmetic-intensive applications [6], and that fracturable LUTs contribute to area efficiency gains [2]. The power analysis indicates that logic and signal activities are the dominant contributors to dynamic power, suggesting that future optimizations should focus on reducing switching activity in the arithmetic units and routing fabric. The use of carry look-ahead adders inherently reduces the number of transitions required for carry propagation compared to ripple-carry designs, which partially explains the favorable power characteristics observed. The results also highlight the importance of considering complete application circuits rather than isolated benchmarks when evaluating arithmetic architectures, as the interactions between components can significantly affect overall performance. The proposed design achieves a balance between the competing demands of speed, area, and power that makes it suitable for portable multimedia and communication applications where energy efficiency is a primary concern. The synthesis and implementation flow using Vivado 2019.1 proved effective for targeting the 22-nanometer FPGA architecture, and the methodology can be readily adapted to other process nodes and device families. The results establish a foundation for further optimization through pipelining, resource sharing, and architectural refinements that could yield additional improvements in the area-delay-power tradeoff.

4. CONCLUSION

The goal of this work was to see whether a carry look-ahead adder, paired with fracturable look-up tables, could make FIR filters faster and more power-efficient on an FPGA. A four-bit CLA was built using six-input LUTs that can also be split into two five-input functions. This mattered because carry logic for different bit positions does not always need the same number of inputs, and the fracturable structure let us use the available logic without wasting it. The filter was coded, simulated, synthesized, and implemented with Vivado 2019.1 on a 22-nm architecture. Behavioral and post-placement simulations both gave the expected filtered output. Timing closure was met, and resource use stayed reasonable across LUTs, flip-flops, and DSP blocks. The power report showed that most dynamic power came from logic and signal switching, while the DSP blocks used much less. Compared with a conventional ripple-carry design, the proposed filter had a shorter critical path and a better area-delay product. These gains were not huge, but they were consistent, and they suggest that combining hardened carry arithmetic with fracturable LUTs is a sensible option for portable DSP hardware. The design still has room to grow. Pipelining, sharing hardware among constant multiplications, and testing on other process nodes could push the power and delay numbers further down. For now, the results support the idea that small architectural choices in the adder and logic block can add up to meaningful improvements in real FIR filter implementations.

DECLARATIONS

Ethical Approval

We affirm that this manuscript is an original work, has not been previously published, and is not currently under consideration for publication in any other journal or conference proceedings. All authors have reviewed and approved the manuscript, and the order of authorship has been mutually agreed upon.

Funding

Not applicable

Availability of data and material

All of the data obtained or analyzed during this study is included in the report that was submitted.

Conflicts of Interest

The authors declare that they have no financial or personal interests that could have influenced the research and findings presented in this paper. The authors alone are responsible for the content and writing of this article.

Authors' contributions

All authors contributed equally in the preparation of this manuscript.

REFERENCES

- [1] Janwadkar, S. and Dhavse, R., **2024**. ASIC design of power and area efficient programmable FIR filter using optimized Urdhva-Tiryagbhyam Multiplier for impedance cardiography. *Microprocessors and Microsystems*, 107, pp.105048. <https://doi.org/10.1016/j.micpro.2024.105048>.
- [2] Murray, K.E., Luu, J., Walker, M.J., McCullough, C., Wang, S., Huda, S., Yan, B., Chiasson, C., Kent, K.B., Anderson, J. and Rose, J., **2020**. Optimizing FPGA logic block architectures for arithmetic. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 28(6), pp.1378-1391. <https://doi.org/10.1109/TVLSI.2020.2965772>.
- [3] Khan, A., Chakraborty, A., Joy, U.B., Wairya, S. and Hasan, M., **2023**. Carry look-ahead and ripple carry method based 4-bit carry generator circuit for implementing wide-word length adder. *Microelectronics Journal*, 140, p.105949. <https://doi.org/10.1016/j.mejo.2023.105949>.
- [4] Kumm, M., Volkova, A. and Filip, S.I., **2022**. Design of optimal multiplierless FIR filters with minimal number of adders. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 42(2), pp.658-671. <https://doi.org/10.1109/TCAD.2022.3179221>.
- [5] Banumathi, J. and Karthy, G., **2026**. High-performance FIR filter designs using Brent Kung Adder and pipelined Vedic multiplier. *Integration*, p.102645. <https://doi.org/10.1016/j.vlsi.2025.102645>.
- [6] Murray, K.E., Luu, J., Walker, M.J., McCullough, C., Wang, S., Huda, S., Yan, B., Chiasson, C., Kent, K.B., Anderson, J. and Rose, J., **2020**. Optimizing FPGA logic block architectures for arithmetic. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 28(6), pp.1378-1391. <https://doi.org/10.1109/TVLSI.2020.2965772>.
- [7] Prasanna, S.C., James, B.P. and Dhandapani, V., **2024**. High speed and area efficient coded input BCSM shared LUT-based FIR filter architecture. *International Journal of System Assurance Engineering and Management*, 15(7), pp.3027-3038. <https://doi.org/10.1007/s13198-024-02312-z>.
- [8] Priya, S.S.S., Karthigaikumar, P. and Teja, N.R., **2022**. FPGA implementation of AES algorithm for high speed applications. *Analog integrated circuits and signal processing*, 112(1), pp.115-125. <https://doi.org/10.1007/s10470-021-01959-z>.
- [9] Raouf, M. and Timarchi, S., **2023**. Non-volatile and high-performance cascadable spintronic full-adder with no sensitivity to input scheduling. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 70(6), pp.2236-2240. <https://doi.org/10.1109/TCSII.2023.3237855>.
- [10] Al Share, A., Zghoul, F.N., Al-Khaleel, O., Al-Khaleel, M. and Papachristou, C., **2023**. Design of high speed BCD adder using CMOS technology. *IEEE Access*, 11, pp.141628-141639. <https://doi.org/10.1109/ACCESS.2023.3342328>.
- [11] Hanuman, C.R.S., Kamala, J. and Aruna, A.R., **2019**. Implementation of multi-precision floating point divider for high speed signal processing applications: CRS Hanuman et al. *The Journal of Supercomputing*, 75(9), pp.6038-6054. <https://doi.org/10.1007/s11227-019-02902-w>.
- [12] Sun, Z., Zhou, Z. and Fu, F.W., **2024**. Optimizing code allocation for hybrid on-chip memory in IoT systems. *Integration*, 97, p.102195. <https://doi.org/10.1016/j.vlsi.2024.102195>.
- [13] Zhang, L., Rao, C. and Lou, X., **2023**. Low-power reconfigurable FIR filter design based on common operation sharing. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 70(8), pp.3169-3173. <https://doi.org/10.1109/TCSII.2023.3256446>.

- [14] Swetha, S. and Siva Sankara Reddy, N., **2024**. Design of FIR filter using low-power and high-speed carry select adder for low-power DSP applications. *IETE Journal of Research*, 70(4), pp.4083-4096. <https://doi.org/10.1080/03772063.2023.2196965>.
- [15] Jacko, P., Duranka, P. and Varga, R., **2024**. Advantages of bistable microwires in digital signal processing. *Sensors*, 24(8), p.2423. <https://doi.org/10.3390/s24082423>.
- [16] Kumar, G.K., Akurati, R.R., Reddy, V.H.P., Cheemalakonda, S., Chagarlamudi, S., Dasari, B. and Shaik, S.S., **2023**. Area-, power-, and delay-optimized 2D FIR filter architecture for image processing applications. *Circuits, Systems, and Signal Processing*, 42(2), pp.780-800. <https://doi.org/10.1007/s00034-022-02232-y>.
- [17] Savalam, C. and Alapati, V.N.T., **2023**. Digital twin based FPGA implementation of FIR filter for multi-bit soft computing error detection and correction for industrial applications: C. Savalam et al. *Soft Computing*, 27(7), pp.4289-4306. <https://doi.org/10.1007/s00500-022-07371-7>.
- [18] Janwadkar, S. and Dhavse, R., **2024**. ASIC implementation of ECG denoising FIR filter by using hybrid Vedic–Wallace tree multiplier. *International Journal of Circuit Theory and Applications*, 52(4), pp.1621-1646. <https://doi.org/10.1002/cta.3844>.
- [19] Sakthimohan, M., Deny, J., Umapathi, K. and Fayek, H.H., **2024**. Enhanced FPGA linear phase FIR filter with amalgam multiplier. *International Journal of Electronics*, 111(10), pp.1653-1678. <https://doi.org/10.1080/00207217.2024.2349972>.
- [20] Rai, A., Roy, A., Qamar, S., Saif, A.G., Hamoda, M.M., Azeem, A. and Mohammed, S.A., **2024**. RETRACTED ARTICLE: Modeling and simulation of FIR filter using distributed arithmetic algorithm on FPGA. *Multimedia Tools and Applications*, 83(31), pp.75855-75868. <https://doi.org/10.1007/s11042-024-18637-7>.