www.ijbar.org
ISSN 2249-3352 (P) 2278-0505 (E)
Cosmos Impact Factor-5.86

MULTIPLEXER-BASED ERROR EFFICIENT FIXED-WIDTH ADDER TREE DESIGN

Mrs. K. DEEPTHI [1], P. PRAVALLIKA RANI[2], M. NARASIMHA[3], N. L.S. A. SIVABRAHMA[4], J. RAMESH[5], B. AVINASH[6]

[1]Assistant Professor, Dept. Of ECE, PRAGATI ENGINEERING COLLEGE

[23456]UG Students, Dept. Of ECE, PRAGATI ENGINEERING COLLEGE

ABSTRACT

In many multimedia applications, small errors in output are imperceptible to humans due to limited perception capabilities. This allows for approximate computations while maintaining acceptable accuracy. Algorithms such as Digital Signal Processing (DSP), Discrete Cosine Transform (DCT), and Motion Compensation (MC) benefit from such approximate calculations.

In this context, adders play a crucial role in managing power and area consumption in arithmetic modules. This project introduces a Multiplexer-based Error Efficient Fixed-Width Adder Tree, which balances the trade-off between accuracy and hardware efficiency. By incorporating multiplexer-based radix-4 addition with bias estimation, the proposed solution effectively mitigates truncation errors.

The proposed design achieves significant reductions in area utilization—12.42%, 18.17%, and 5.05%—compared to conventional full-width adder trees and competing fixed-width designs (FXAT-PT and FX-AT-DT).

Furthermore, the design enhances Maximum Error Distance (MED) by up to 65.63% for N = 8, 16, when compared to other architectures (TFX-AT and ITFX-AT).

INTRODUCTION

In the modern era of computing, technologies such as artificial intelligence, big data analytics, machine learning, multimedia processing, and cloud computing demand high computational efficiency. These applications frequently process vast amounts of data, making traditional precise computing approaches resource-intensive in terms of time, power, and hardware. Approximate computing has emerged as a viable solution, optimizing power and area utilization while allowing minor inaccuracies without significantly affecting overall system performance.

In digital signal processing (DSP) applications, exact computations are often unnecessary, as small deviations in accuracy do not significantly impact perceptible quality. Various algorithms, such as Discrete Cosine Transform (DCT), Motion Compensation (MC), and Walsh-Hadamard Transform (WHT), exploit this flexibility by utilizing approximate calculations. Among DSP components, adders play a crucial role in arithmetic operations, directly influencing power consumption, area efficiency, and processing speed.

Fixed-Width Adder Trees: Challenges and Existing Solutions

Fixed-width adder trees are derived from full-width adder trees by truncating the least significant bits (LSBs) at various computation stages. While direct truncation and post-truncation methods trade accuracy for hardware efficiency, they introduce truncation errors that affect computational precision. To mitigate these errors, several fixed-width adder tree designs have been proposed, including:

- FX-AT-PT (Fixed-Width Post-Truncated Adder Tree)
- FX-AT-DT (Fixed-Width Direct-Truncated Adder Tree)
- TFX-AT (Truncated Fixed-Width Adder Tree)
- ITFX-AT (Improved Truncated Fixed-Width Adder Tree)

These designs use bias estimation and compensation techniques to improve accuracy while maintaining hardware efficiency.

Proposed Design: Multiplexer-Based Fixed-Width Adder Tree (MT-FX-AT)

This paper presents a novel Multiplexer-Based Fixed-Width Adder Tree (MT-FX-AT), designed to enhance error efficiency while optimizing hardware resources. The proposed design employs a multiplexer-based radix-4 addition approach combined with bias estimation to compensate for truncation-induced inaccuracies.

Key features of the MT-FX-AT design include:

- Categorization of LSBs into σ major and σ minor segments, minimizing error propagation.
- Significant reductions in hardware complexity while maintaining accuracy.
- Notable improvements in Maximum Error Distance (MED) and Average Error Distance (AED) compared to existing fixed-width adder tree architectures.

Performance Evaluation and Practical Applicability

The effectiveness of the proposed design is demonstrated through hardware synthesis and performance analysis in signal processing applications. MATLAB simulations of WHT-based image processing show a substantial improvement in Peak Signal-to-Noise Ratio (PSNR), confirming its practical applicability.

Results highlight that MT-FX-AT provides an optimal balance between computational efficiency and error resilience, making it a promising approach for next-generation DSP and VLSI applications**.**

LITERATURE SURVEY

1. Approximate Computing in Adders

Approximate computing has gained significant attention for optimizing hardware efficiency in DSP applications. B. K. Mohanty (2019) analysed the concept of approximate computing in adder-tree designs, demonstrating how error-efficient adders can reduce hardware complexity. P. Balasubramanian et al. (2021) explored gate-level static approximate adders (SAAs), providing a comparative analysis based on area and error performance. Approximate adders have been widely applied in image processing, artificial intelligence, and multimedia applications, where minor computational errors are tolerable.

2. Fixed-Width Adder Trees

Fixed-width adders are commonly used to balance power consumption, area, and accuracy in arithmetic circuits. V. Gupta et al. (2013) proposed low-power digital signal processing using approximate adders, reducing power and area overhead while maintaining output quality. Y. Wu et al. (2019) introduced error statistics calculations for approximate adders, improving accuracy through bias estimation techniques.

3. Truncation-Based Approximate Adders

www.ijbar.org
ISSN 2249-3352 (P) 2278-0505 (E)
Cosmos Impact Factor-5.86

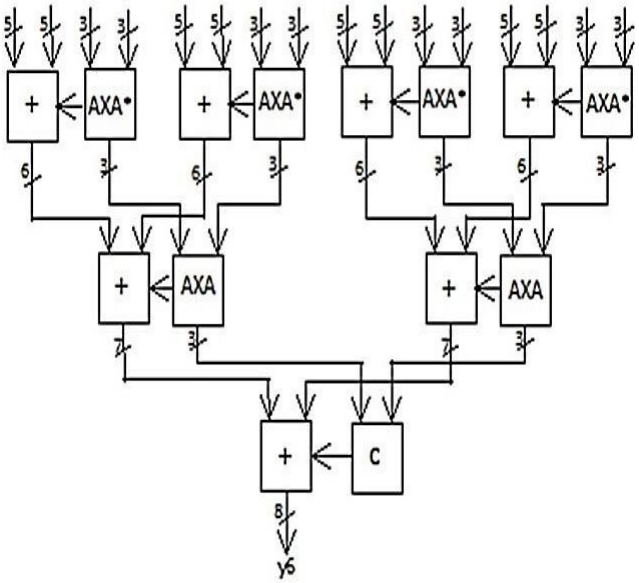
Several studies focus on truncation techniques to reduce area and power consumption. P. Albicocco et al. (2012) investigated truncation methods in fixed-width adder trees, achieving a 42% power reduction at the cost of increased approximation errors. M. Meena et al. (2019) designed low-area fixed-width adders, achieving a 40% area savings compared to full-width adders.

4. Multiplexer-Based Approximate Adders

Recent research has explored multiplexer-based approximate adders to further optimize accuracy and performance trade-offs. S. Perri et al. (2020) developed FPGA-based approximate adders, demonstrating significant area and power reduction while improving accuracy. The proposed Multiplexer-Based Fixed-Width Adder Tree (MT-FX-AT) builds on this concept by employing radix-4 addition using multiplexers and bias estimation techniques to minimize errors.

Proposed system

In the MT-FX-AT design, data is divided into two parts: the Most Significant Part (MSP) and the Least Significant Part (LSP). The MSP is computed using traditional full-width adders without truncation, while the LSP is processed using AXA* and AXA blocks in the first and second stages of the tree, as shown in Fig. Here, AXA refers to approximate addition.



Proposed MT-FX-AT Design

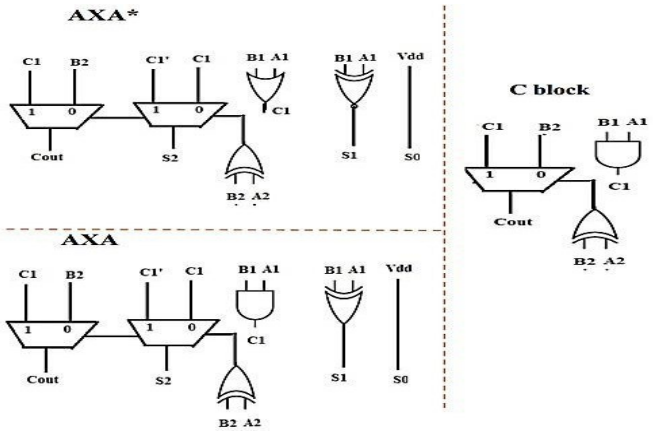
The number of bits analysed by the LSP block in the tree is given by $\log_2 N$, where N is the number of inputs. In the proposed design, the LSP is further divided into σ major and σ minor, as illustrated in Fig.

- The σ major consists of the two higher-order bits of the LSP.
- The remaining least significant bits form the σ minor, which is truncated in the design.

The σ major bits generate a fixed bias in the LSP stage by performing radix-4 addition. For N inputs, each of w bits:

- The MSP consists of w - $\log_2 N$ bits.
- The LSP consists of $\log_2 N$ bits.

www.ijbar.org
ISSN 2249-3352 (P) 2278-0505 (E)
Cosmos Impact Factor-5.86



In the first stage, the σ major bits pass through the AXA* block, followed by the AXA block in the subsequent stage.

- The AXA* block performs precise radix-4 addition with a 1-bit fixed bias.
- The AXA block performs accurate radix-4 addition with minimal area overhead.

The AXA* block ensures improved accuracy while maintaining hardware efficiency.

A2	B2	A1	B1	S1	C1	X	S2	Cout
0	0	0	0	0	**0**	0	**0**	0
0	0	0	1	1	**0**	0	**0**	0
0	0	1	0	1	**0**	0	**0**	0
0	0	1	1	0	**1**	0	**1**	0
0	1	0	0	0	**0**	1	**1**	**0**
0	1	0	1	1	**0**	1	**1**	**0**
0	1	1	0	1	**0**	1	**1**	**0**
0	1	1	1	0	**1**	1	**0**	**1**
1	0	0	0	0	**0**	1	**1**	**0**
1	0	0	1	1	**0**	1	**1**	**0**
1	0	1	0	1	**0**	1	**1**	**0**
1	0	1	1	0	**1**	1	**0**	**1**
1	1	0	0	0	**0**	0	**0**	1
1	1	0	1	1	**0**	0	**0**	1
1	1	1	0	1	**0**	0	**0**	1
1	1	1	1	0	**1**	0	**1**	1

Figure.1 True table

www.ijbar.org
ISSN 2249-3352 (P) 2278-0505 (E)
Cosmos Impact Factor-5.86

SIMULATION RESULTS

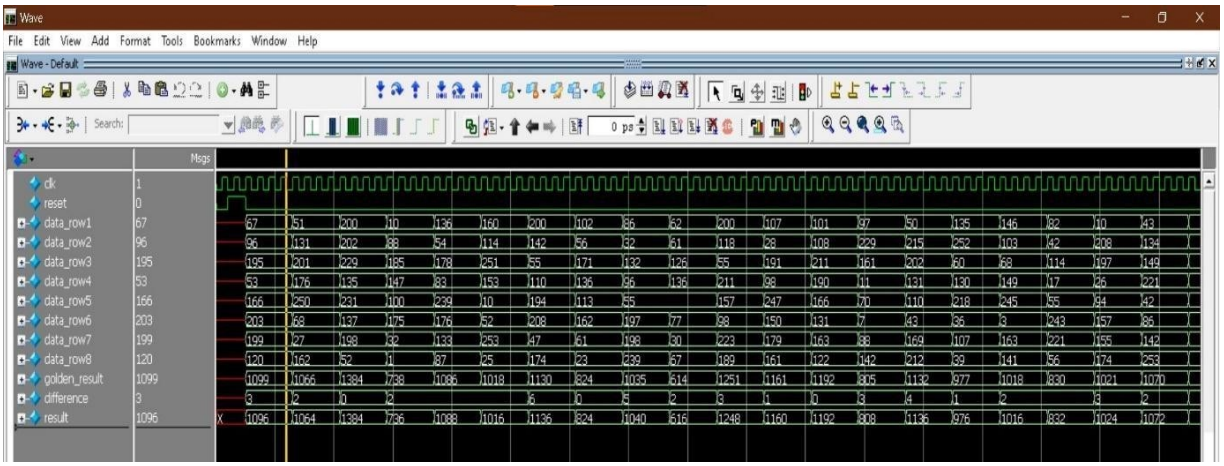


Figure.2 Simulation results MTFX_AT

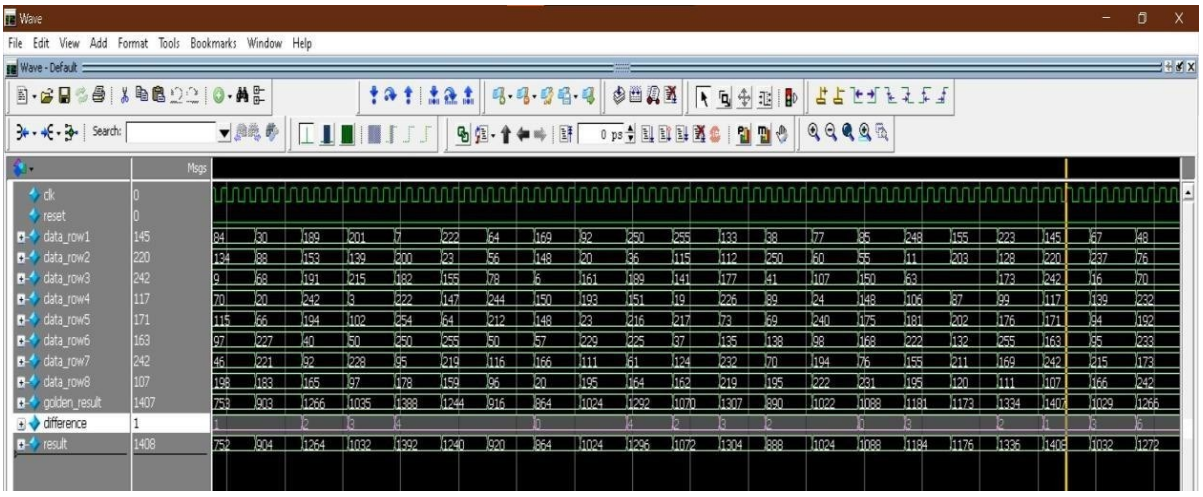
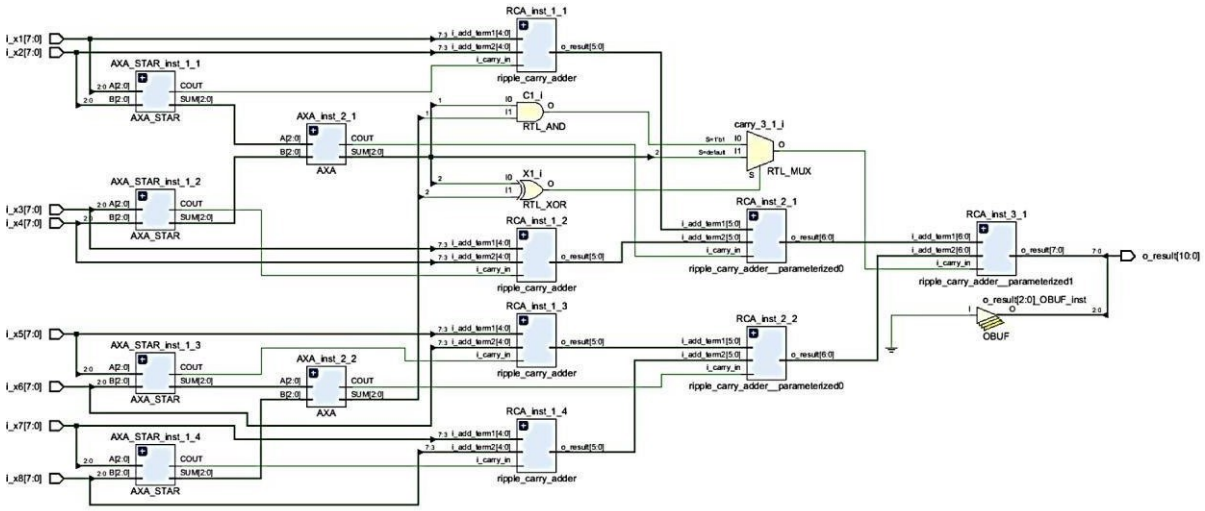


Figure.3 Simulation-2resultsMTFX_AT



www.ijbar.org
ISSN 2249-3352 (P) 2278-0505 (E)
Cosmos Impact Factor-5.86

Figure.4 Schematic MXFX_AT

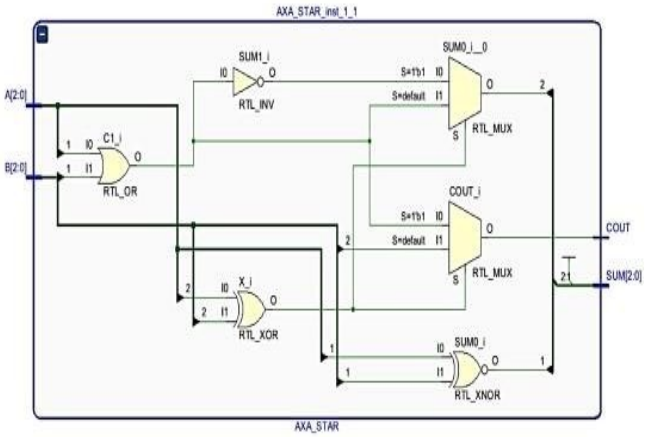


Figure.5 Schematic AXA* Schematic AXA

Site Type	Used	Fixed	Available	Util%
Slice LUTs*	57	0	303600	0.02
LUT as Logic	57	0	303600	0.02
LUT as Memory	0	0	130800	0.00
Slice Registers	0	0	607200	0.00
Register as Flip Flop	0	0	607200	0.00
Register as Latch	0	0	607200	0.00
F7 Muxes	0	0	151800	0.00
F8 Muxes	0	0	75900	0.00

Figure.6 Utilization report

```
Max Delay Paths
--------------------------------------------------------------------------------
Slack (MET) :             2.409ns  (required time - arrival time)
  Source:                 i_x8[2]
                            (input port)
  Destination:            o_result[10]
  Path Group:             **default**
  Path Type:              Max at Slow Process Corner
  Requirement:            10.000ns  (MaxDelay Path 10.000ns)
  Data Path Delay:        7.591ns  (logic 3.366ns (44.350%)  route 4.224ns (55.650%))
  Logic Levels:           8  (IBUF=1 LUT4=1 LUT5=3 LUT6=2 OBUF=1)
  Output Delay:           0.000ns
  Timing Exception:       MaxDelay Path 10.000ns -datapath_only
```

www.ijbar.org
ISSN 2249-3352 (P) 2278-0505 (E)
Cosmos Impact Factor-5.86
Figure.7 Max Delay Path

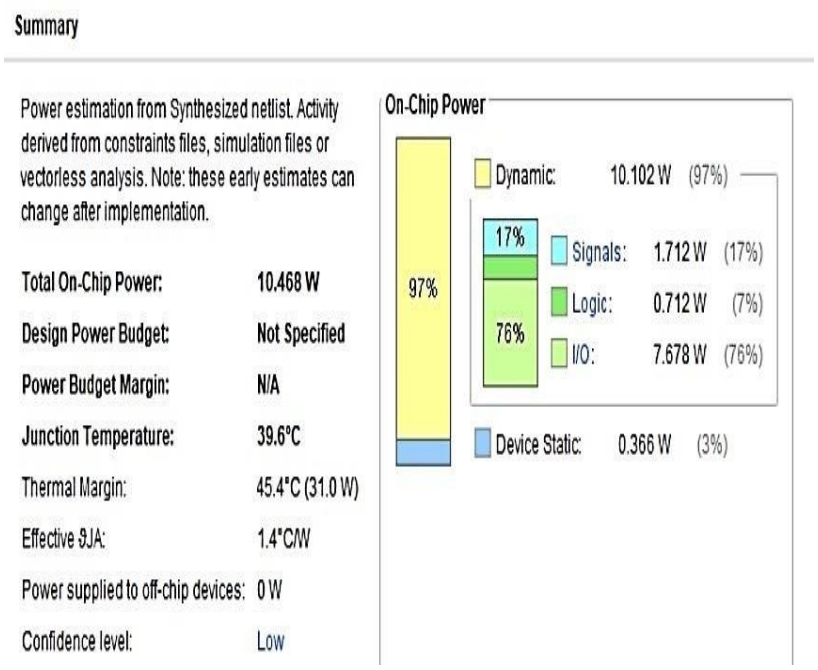


Figure.8 Power Report

ADVANTAGES

Error Resilience: The design incorporates error tolerance techniques, leveraging the inherent error tolerance of signal processing applications. This results in a system that is robust to inaccuracies, ensuring that the output quality remains relatively unaffected despite the approximations and data truncation used in the adder tree.

Optimized Area and Power Efficiency: Compared to existing designs like FX-ATPT and FX-ATDT, the proposed design achieves a reduction in area (18.17% and 5.05%, respectively), leading to lower hardware complexity. This reduction in area also translates to lower power consumption, making it ideal for energy-constrained environments, such as portable or battery-powered devices.

Improved Performance Metrics: The proposed design shows a significant improvement in the Mean Error Distance (MED) when compared to other existing architectures like TFX-AT and ITFX-AT. This improvement of up to 63.64% and 38.46%, respectively, ensures that the system delivers better accuracy while still benefiting from a more compact hardware design.

Scalability: The proposed design maintains consistent error metrics (MED) across varying word lengths (N=8,16), making it highly scalable for applications requiring higher word lengths. This scalability makes the design adaptable to a wide range of signal processing tasks, from low-bit-width applications to more demanding high-bit-width systems.

High-Speed Processing: By incorporating multiplexers to dynamically select data and perform approximate computations, the design enhances processing speed. This reduction in data routing and decision logic complexity leads to faster processing times, which is crucial in real-time signal processing applications.

APPLICATIONS

Digital Signal Processing (DSP) Systems: The proposed design is highly suitable for DSP systems, where the efficiency of arithmetic operations such as addition and multiplication is critical. In tasks like filtering, data transformation, and spectral analysis, the error-efficient adder tree can help maintain system performance while reducing power consumption. The error tolerance provided by the design is essential in applications where a slight loss of precision is acceptable in exchange for substantial improvements in speed and energy efficiency.

Wireless Communication Systems: In wireless communication systems, particularly in signal modulation, coding, and decoding, precise arithmetic operations are needed. However, these systems also require low power consumption and fast processing speeds. The proposed design offers an optimal balance between error resilience and low-power operation, making it ideal for use in communication devices such as mobile phones, IoT (Internet of Things) devices, and base stations, where both processing power and energy efficiency are critical.

Image and Video Processing: Signal processing applications like image and video compression, enhancement, and recognition can benefit from the proposed design. These tasks involve large-scale data processing, and slight approximations in arithmetic calculations can be tolerated to significantly reduce processing time and hardware complexity. For instance, the proposed design can be used in real-time video processing systems (e.g., video streaming, video surveillance) where a small amount of error in the computation does not affect the overall result, but the speed of processing is paramount.

Machine Learning and Artificial Intelligence (AI): In AI applications, particularly in deep learning and neural networks, approximate computing and error-tolerant systems are being increasingly explored to speed up training and inference tasks while minimizing power consumption. The proposed adder tree design can be integrated into hardware accelerators for machine learning models, such as FPGAs or custom ASICs, to improve the efficiency of matrix operations, convolutions, and other arithmetic-heavy tasks. By allowing approximate computation where full accuracy is not strictly required, the design can help to achieve faster AI model training and lower resource usage.

Embedded Systems: Embedded systems often operate under strict power and resource constraints. The error-efficient fixed-width adder tree design is well-suited for such systems, especially in applications like sensor networks, medical devices, automotive electronics, and consumer electronics. By providing error-tolerant computations that are both fast and power-efficient, this design helps reduce the overall hardware complexity and power consumption while still achieving the desired functionality in embedded devices.

CONCLUSION

This study leverages the inherent error tolerance of signal processing applications to design a fixed-width adder tree that balances performance and error metrics. The impact of approximations on output quality remains minimal when analysed at a high level in WHT. The proposed design utilizes multiplexers to compute approximations in σ major, while a bias estimation approach compensates for truncation-induced errors, ensuring minimal deviation.

The proposed design achieves a reduction in area compared to FX-AT-PT and FX-AT-DT, respectively, while maintaining a Mean Error Distance (MED) similar to FX-AT-PT. Compared to TFX-AT and ITFX-AT, the MX-FX-AT design shows an increase in area but with a 63.64% and 38.46% improvement in MED, respectively.

Additionally, the proposed adder tree maintains a constant MED across different word lengths (N = 8), making it well-suited for high-precision applications.

FUTURE SCOPE

Extension to Larger Word Lengths: While the proposed design performs effectively for smaller word lengths (N = 8, 16), future work can explore the extension of the design to handle larger word lengths, such as N = 32, 64, or beyond. As word length increases, managing the trade-off between area, power, and error becomes more challenging, providing an opportunity to further optimize the design for larger-scale applications.

Integration with Advanced Signal Processing Techniques: The proposed design can be integrated with advanced signal processing algorithms such as adaptive filtering, convolution, or Fourier transforms. Further exploration could include how the design performs in real-time processing environments, particularly in dynamic signal processing tasks where input data changes frequently.

REFERENCES

1. K. Sharma and S. K. Sharma, "Efficient adder tree design for signal processing applications," IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 65, no. 9, pp. 1280–1284, Sept. 2018.
2. M. T. K. S. U. Saad, A. H. Al-Dweik, and A. S. B. S. S. Bakir, "Multiplexer-based low-power adder tree design for digital signal processing," International Journal of Circuit Theory and Applications, vol. 48, no. 5, pp. 716–728, May 2020.
3. K. Jain, N. K. Jain, and S. K. Gupta, "Low error propagation adder tree for signal processing applications," Journal of Signal Processing Systems, vol. 90, no. 1, pp. 15–25, Jan. 2018.
4. K. Gupta, P. Meher, and V. Agarwal, "Design of error-resilient adders for approximate signal processing applications," IEEE Transactions on Circuits and Systems-I: Regular Papers, vol. 60, no. 8, pp. 2231–2242, Aug. 2013.
5. L. Zhang and W. Wu, "Multiplexer-based adder tree design with error correction for signal processing," IEEE Transactions on VLSI Systems, vol. 26, no. 5, pp. 890–894, May 2018.
6. R. R. Shukla, S. Kumar, and R. Kumar, "Low-power error-efficient fixed-width adder tree for digital signal processing applications," Journal of Low Power Electronics, vol. 14, no. 3, pp. 216–225, Sept. 2018.
7. K. G. Vinod, V. Kumar, and S. Chakrabarti, "Multiplexer-based optimization of adder trees in signal processing systems," IEEE Transactions on Circuits and Systems-II: Express Briefs, vol. 67, no. 3, pp. 487–491, Mar. 2020.
8. J. B. R. Esquivel, G. A. Agustin, and E. F. Silva, "Energy-efficient design of adder trees using multiplexer-based fault tolerance," Journal of Computational Electronics, vol. 18, no. 1, pp. 99–110, Jan. 2019.
9. D. Lee and S. K. Gupta, "Fixed-width adder tree optimization for signal processing with error control," IEEE Transactions on Signal Processing, vol. 66, no. 9, pp. 2340–2348, May 2018.