Design of a High-Throughput and Memory-Efficient Radix-4 FFT Architecture

M.Usha[1]
Department of Electronics and Communication Engineering
University College of Engineering and Technology
Acharya Nagarjuna University, Guntur, India
Email: usha.mallaparapu@gmail.com

K .Asritha[2]
Department of Electronics and Communication Engineering
University College of Engineering and Technology
Acharya Nagarjuna University, Guntur, India
Email: asrithakamarajugadda@gmail.com

A.Nagasai[3]
Department of Electronics and Communication Engineering
University College of Engineering and Technology
Acharya Nagarjuna University, Guntur, India
Email:sakula640@gmail.com

B. Harikrishna Naik[4]
Department of Electronics and Communication Engineering
University College of Engineering and Technology
Acharya Nagarjuna University, Guntur, India
Email: harinaik8189@gmail.com

Assistant Professor[1], Students[2,3,4]

***Abstract*—The Fast Fourier Transform (FFT) is a fundamental algorithm in digital signal processing, widely used in communication, radar, biomedical systems, and image processing applications. However, conventional FFT architectures suffer from high latency, increased hardware complexity, and inefficient memory utilization due to repeated computations and multiple processing stages.**

This paper presents a high-throughput and memory-efficient Radix-4 FFT architecture implemented using Verilog HDL. The proposed design reduces computational complexity by decreasing the number of stages and improves throughput using pipelined processing. Additionally, a memory optimization technique based on data reuse is introduced to minimize memory access opera- tions.

The architecture is implemented on FPGA using Xilinx Vivado, and performance is evaluated in terms of delay, throughput, and resource utilization. Results show that the proposed design achieves significant improvements compared to traditional Radix- 2 FFT architectures, making it suitable for real-time high-speed signal processing systems.

***Index Terms*—FFT, Radix-4, FPGA, Verilog HDL, Pipelining, Memory Optimization, High Throughput**

I. Introduction

The Fast Fourier Transform (FFT), introduced by Cooley and Tukey, is one of the most important algorithms in digital signal processing. It is used to convert signals from the time domain to the frequency domain efficiently. The FFT reduces the computational complexity of the Discrete Fourier Transform (DFT) from $O(N^2)$ to $O(N \log N)$, making it highly suitable for real-time applications.

Modern communication systems such as Orthogonal Frequency Division Multiplexing (OFDM), 5G wireless communication, radar signal processing, and image processing require high-speed FFT processors capable of handling large volumes of data with minimal delay.

The DFT computes frequency components of a discrete signal but requires a large number of multiplications and additions, latency, higher power consumption, and inefficient memory utilization.

To address these limitations, Radix-4 FFT architecture is introduced. It processes four inputs simultaneously, reducing the number of stages by half compared to Radix-2 FFT. This results in improved computational efficiency, reduced latency, and enhanced throughput.

The main contributions of this work include:

- Design of a high-throughput Radix-4 FFT architecture
- Implementation of pipelined processing to increase speed
- Optimization of memory usage using data reuse techniques
- FPGA-based implementation and performance evaluation

II. Background

A. Discrete Fourier Transform

The N-point Discrete Fourier Transform (DFT) is defined as:

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk} \quad (1)$$

where $W_N = e^{-j\frac{2\pi}{N}}$ is the twiddle factor

Conventional Radix-2 FFT architectures are widely used due to their simple implementation. However, they require a large number of computational stages, leading to increased

B. Radix-4 FFT Algorithm

Radix-4 FFT improves efficiency by decomposing the DFT into smaller parts. Instead of dividing the sequence into two parts (Radix-2), it divides into four parts, reducing the number of stages.

The Radix-4 butterfly equations are:

$$Y_0 = x_0 + x_1 + x_2 + x_3 \quad (2)$$

$$Y_1 = x_0 - jx_1 - x_2 + jx_3 \quad (3)$$

$$Y_2 = x_0 - x_1 + x_2 - x_3 \quad (4)$$

$$Y_3 = x_0 + jx_1 - x_2 - jx_3 \quad (5)$$

This reduces computational complexity and improves performance.

III. Proposed Architecture

A. System Overview

The proposed Radix-4 FFT architecture is designed to achieve high throughput and efficient memory utilization by integrating multiple functional modules in a structured and synchronized manner. The architecture follows a modular design approach where each block performs a specific function, enabling parallelism and scalability.

The overall system consists of four primary components: input buffering unit, Radix-4 butterfly processing unit, pipeline stages, and output generation block. These components work together to ensure continuous data flow and efficient compu- tation.

The input buffering unit is responsible for collecting incom- ing data samples from the input stream. Since FFT requires a fixed number of input samples (typically powers of 4 in Radix-4), the buffer temporarily stores incoming data until a complete set of inputs is available. This buffering mechanism ensures proper alignment and synchronization of input data for subsequent processing stages.

The buffered inputs are then fed into the butterfly processing unit, which performs the core FFT computations. This unit executes complex arithmetic operations such as addition, sub- traction, and multiplication with twiddle factors. By adopting Radix-4 decomposition, the architecture processes four input samples simultaneously, reducing the number of computational stages compared to Radix-2 FFT.

To further enhance performance, the architecture incorporates pipelining. The pipeline stages divide the computation into smaller sub-operations, allowing multiple data sets to be processed concurrently. This significantly increases throughput and ensures that the system can handle high-speed data streams without bottlenecks.

The output generation block collects the processed data from the pipeline stages and formats it into the final FFT output sequence. It ensures that the output is correctly ordered and synchronized with the control signals.

Overall, the system architecture is optimized to balance speed, hardware complexity, and memory usage, making it suitable for real-time signal processing applications such as wireless communication and image processing.

B. Radix-4 Butterfly Unit

The Radix-4 butterfly unit forms the fundamental building block of the proposed FFT architecture. It is responsible for performing the core mathematical operations required for transforming input signals from the time domain to the frequency domain.

In Radix-4 FFT, the input sequence is divided into groups of four samples, and each group is processed simultaneously. This reduces the total number of stages required for compu- tation, thereby decreasing latency and improving efficiency.

Let the four inputs to the butterfly unit be denoted as x0, x1, x2, x3 The butterfly computation generates four output values through a combination of additions, subtractions, and complex multiplications. The mathematical operations involved can be represented as: y0 = x0 + x1 + x2 + x3 y1 = x0 - jx1 - x2 + jx3 y2 = x0 - x1 + x2 - x3 y3 = x0 + jx1 - x2 - jx3 These equations demonstrate that the butterfly operation efficiently combines input samples to produce intermediate frequency components. The use of complex arithmetic allows accurate representation of both magnitude and phase information.

One of the key advantages of the Radix-4 butterfly is the reduction in the number of required multiplications compared to Radix-2. Since multipliers consume significant hardware resources, this reduction leads to improved area efficiency and lower power consumption.

Additionally, the butterfly unit is designed to operate in parallel, enabling simultaneous processing of multiple data sets. This parallelism is crucial for achieving high throughput in real-time applications.

The implementation of the butterfly unit in Verilog involves designing combinational logic for arithmetic operations and registers for storing intermediate results. Careful optimization is required to minimize propagation delay and ensure stable operation at high clock frequencies.

C. Pipeline Architecture

Pipelining is a key technique used in the proposed FFT architecture to enhance performance and throughput. It involves dividing the overall computation into multiple stages, where each stage performs a specific portion of the operation.

In the absence of pipelining, FFT computation would be performed sequentially, leading to increased latency and inefficient utilization of hardware resources. By introducing pipeline stages, multiple input data sets can be processed simultaneously at different stages of computation.

The proposed design employs a two-stage pipeline architecture. In the first stage, intermediate butterfly computations are performed. This stage processes the input samples and generates partial results. These results are then stored in pipeline registers.

In the second stage, final output computations are carried out using the intermediate values from the first stage. This stage completes the FFT transformation and produces the final frequency-domain outputs.

Pipeline registers play a crucial role in maintaining synchronization between stages. They store intermediate values and ensure that data is correctly transferred from one stage to the next at each clock cycle. These registers also help in reducing timing issues such as race conditions and data hazards.

The use of pipelining significantly improves throughput, as a new input can be fed into the system at every clock cycle once the pipeline is filled. This makes the architecture suitable for high-speed applications where continuous data processing is required.

However, pipelining introduces additional hardware overhead due to the use of registers. Therefore, a balance must be maintained between performance improvement and hardware complexity.

The proposed design includes:

- Stage 1: Intermediate butterfly computation
- Stage 2: Final output calculation

Pipeline registers store intermediate values between stages, ensuring proper synchronization.

D. Memory Optimization

Memory optimization is a critical aspect of FFT architecture design, as memory usage directly impacts hardware cost, power consumption, and overall system performance.

In conventional FFT implementations, multiple memory accesses are required at each stage of computation. This leads to increased latency and higher power consumption due to frequent read and write operations.

The proposed architecture addresses these challenges by employing efficient memory optimization techniques. One of the primary strategies is data reuse, where intermediate results are reused instead of recomputing them. This reduces the number of arithmetic operations and minimizes memory access.

Another important technique is register-based storage. Instead of using large memory blocks such as RAM, the design utilizes registers to store intermediate values. Registers offer faster access times and lower latency compared to memory blocks, making them suitable for high-speed applications.

The architecture also minimizes redundant memory access by carefully organizing data flow. By ensuring that data is accessed only when necessary, unnecessary read and write operations are avoided.

Additionally, the use of efficient data scheduling techniques ensures that data is processed in an optimal sequence, reducing memory bottlenecks and improving overall performance.

These memory optimization strategies result in reduced hardware complexity, lower power consumption, and improved processing speed, making the architecture efficient

E. Control Logic

Control logic plays a vital role in coordinating the operation of different components within the FFT architecture. It ensures that data flows smoothly between modules and that computations are performed in the correct sequence.

IV. Implementation

The proposed Radix-4 FFT architecture is implemented using Verilog Hardware Description Language (HDL), which provides a robust platform for designing and model digital systems at the Register Transfer Level (RTL). The choice of Verilog enables precise control over data flow, timing, and hardware resources, making it highly suitable for implementing complex signal processing algorithms such as FFT. The design is synthesized and simulated using the Xilinx Vivado Design Suite, which supports FPGA-based development and provides detailed reports on performance metrics such as timing, power, and resource utilization.

The implementation begins with the transformation of the mathematical Radix-4 FFT algorithm into a hardware-efficient structure. Unlike software implementations that rely on sequential execution, hardware design focuses on parallelism and concurrency. The Radix-4 decomposition naturally supports parallel processing by grouping input samples into sets of four, thereby reducing the number of computational stages required. This structural advantage is effectively utilized in the hardware design to achieve high throughput.

The architecture is divided into several modules, each responsible for a specific function within the system. These modules include the butterfly computation unit, pipeline register unit, control logic unit, and input-output interface unit. The modular design approach allows independent development and testing of each block, ensuring correctness and simplifying debugging.

The butterfly computation unit forms the computational core of the design. It is responsible for performing complex arithmetic operations including addition, subtraction, and multiplication with twiddle factors. In the proposed design, fixed-point arithmetic is used instead of floating-point representa- tion to reduce hardware complexity and power consumption. The use of fixed-point representation requires careful scaling and normalization to maintain numerical accuracy. Twiddle factor multiplication, which is typically resource-intensive, is optimized using constant coefficient multipliers and shift-add techniques. In certain cases, multiplications involving imaginary units (±j) are simplified to sign changes and swapping operations, further reducing hardware requirements.

The pipeline register unit is introduced to enhance throughput by enabling concurrent execution of multiple stages of computation. Pipeline registers are inserted between different stages of the FFT processing chain to store intermediate results. This allows new input data to enter the system before the previous data has fully completed processing, effectively overlapping computations. The pipeline is carefully designed to balance delays across stages, ensuring that no stage becomes a valid in and valid out which indicates the presence of valid input data and output data.

Functional verification of the design is carried out using simulation waveforms generated in Vivado. A testbench is de- bottleneck. Proper synchronization is maintained using clock signals, and timing constraints are applied during synthesis to achieve stable operation at high frequencies.

The control logic unit coordinates the operation of all modules within the architecture. It generates control signals such

veloped to apply various input signals to the design, including both random inputs and known test cases with predictable FFT outputs. The simulation results are analyzed using waveform viewers to verify the correctness of the output and the proper

functioning of pipeline stages. Particular attention is given to timing relationships between signals, ensuring that data is correctly propagated through the pipeline without loss or corruption.

V. Results and Analysis

The performance of the proposed Radix-4 FFT architecture is evaluated using synthesis and simulation reports obtained from the Xilinx Vivado Design Suite. The evaluation focuses on key performance parameters including latency, throughput, hardware resource utilization, and power consumption. These metrics provide a comprehensive understanding of the efficiency and practicality of the proposed design.

Latency is defined as the time required for the system to produce output after receiving input. In conventional Radix- 2 FFT architectures, the number of computational stages is relatively high, leading to increased latency. In contrast, the Radix-4 FFT reduces the number of stages by a factor of two, as it processes four inputs simultaneously. This reduction in stages directly translates to lower latency. Furthermore, the use of pipelining allows overlapping of computations, which effectively reduces the perceived latency once the pipeline is filled.

Throughput is a measure of how many output samples are produced per unit time. The proposed architecture achieves high throughput due to its pipelined design. Once the pipeline reaches steady-state operation, a new output is generated at every clock cycle. This continuous flow of data makes the architecture highly suitable for real-time applications where large volumes of data must be processed quickly. The throughput is further enhanced by the parallel processing capabilities of the Radix-4 butterfly unit.

Hardware resource utilization is analyzed in terms of FPGA components such as Look-Up Tables (LUTs), Flip-Flops (FFs), Block RAM (BRAM), and DSP slices. The proposed de- sign demonstrates efficient utilization of these resources by minimizing the number of multipliers and reducing memory requirements through data reuse techniques. The use of registers instead of large memory blocks for storing intermediate results contributes to lower latency and reduced resource consumption.

Power consumption is another critical parameter, especially for portable and embedded systems. The proposed architecture reduces power consumption by minimizing switching activity and reducing the number of active components. The reduction in computational stages and memory access operations further contributes to energy efficiency.

Simulation results confirm the correctness and stability of the design under various input conditions. The output wave- forms match expected FFT results, and no timing violations are observed during synthesis. The design operates reliably at the target clock frequency, demonstrating its suitability for high-speed applications Compared to conventional Radix-2 FFT:

The number of stages is reduced,resulting in lower latency

- Pipelining enables higher throughput
- Memory usage is minimized through efficient data reuse

Simulation results confirm that the proposed design operates correctly and efficiently under different input conditions.

VI. Experimental Outputs

This section presents the practical outputs obtained from the FPGA implementation of the proposed Radix-4 FFT architecture. The results include dataflow design, dataflow diagram, implementation structure, resource utilization, and power analysis.

A. A. Simulation Waveforms

Simulation waveforms are used to verify the functional correctness of the design. Fig. 1 shows the input-output relationship along with control signals such as valid_in and valid out.

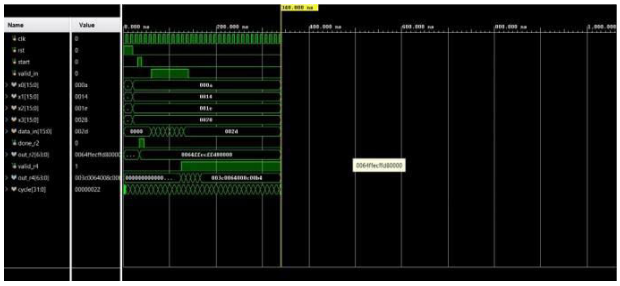

Fig. 1. Simulation Waveforms

Initially, the reset signal is asserted to clear all internal registers. After the reset is deactivated, input data is applied sequentially at each clock cycle. The FFT computation begins, and due to the pipelined architecture, the output is produced after a certain delay known as latency.The waveform clearly shows that the output does not appear immediately after the input is applied. Instead, it is delayed by a few clock cycles, which confirms the pipelined operation of the design. Once the computation is complete, the valid signal becomes high, indicating that the output values are correct.

B. Dataflow Design

The dataflow design of the proposed Radix-4 FFT architec- ture is shown in Fig. 2. It illustrates how input data propagates through different functional blocks such as buffering units, butterfly processors, and pipeline stages.

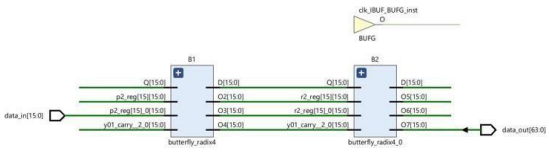

Fig. 2. Dataflow Design of Radix-4 FFT

C. Synthesized Design

The synthesized design shown in Fig. 3 provides a synthesis converts Verilog code (RTL) into a hardware-level design that can be implemented on FPGA.

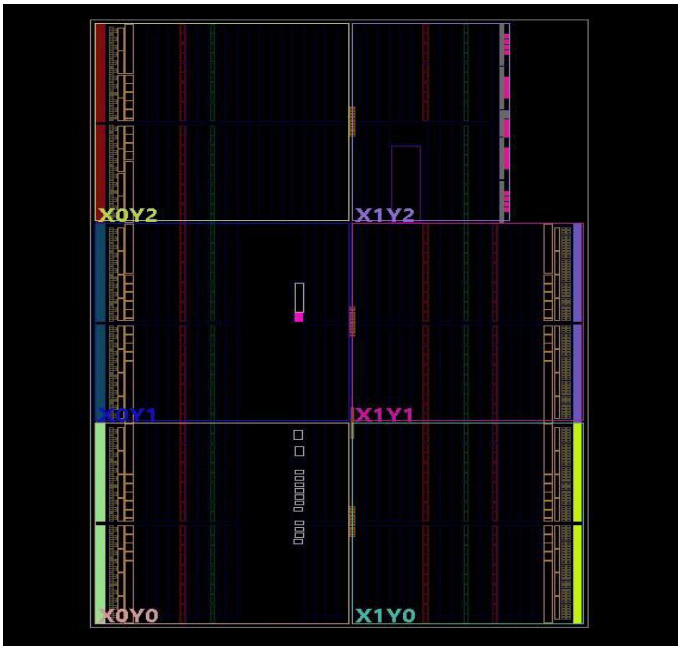

Fig. 3 synthesized Design

The FFT design is mapped into hardware blocks. The input data flows through registers and arithmetic units. The butterfly units perform required computations, while pipeline registers store intermediate outputs. The control logic ensures proper timing and synchronization.

D.Radix-4 FFT Implementation

The implementation of the proposed Radix-4 FFT architec- ture is shown in Fig. 4.

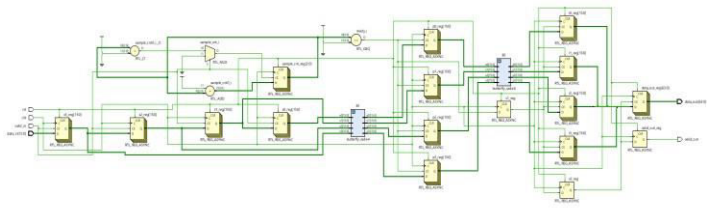

Fig. 4. Radix-4 FFT Implementation

The schematic diagram represents the hardware implementation of the Radix-4 FFT architecture. It shows how different modules such as memory, butterfly units, and pipeline registers are interconnected to perform FFT computation efficiently

E .Resource Utilization

The FPGA resource utilization report is shown in Fig. 5.

Resource	Utilization	Available	Utilization %
LUT	224	20800	1.08
FF	262	41600	0.63
IO	84	106	79.25

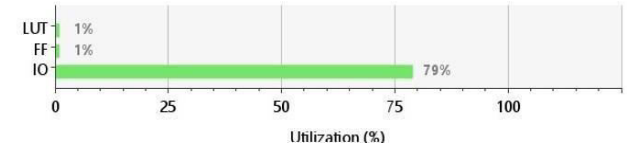

Fig. 5. Resource Utilization Report

The table quantifies exactly how many hardware "building blocks" your design uses compared to what is physically available on the chip.

LUTs: These are the primary logic elements used to implement combinational logic. Using only 1.08% indicates the logic complexity of your design is very low for this specific chip.

FFs: These provide the memory/registers for sequential logic. A 0.63% usage shows very little data storage or state-machine complexity is required.

IOs: These represent the physical pins used to communicate with the outside world. At 79.25%, zthis is your most heavily used resource.

BUFG: These are high-fanout buffers used typically for clock signals. Using 3% suggests you have a very simple clocking scheme (likely just one or two primary clocks).

F.On-Chip Power Analysis

The on-chip power analysis is shown in Fig. 6.

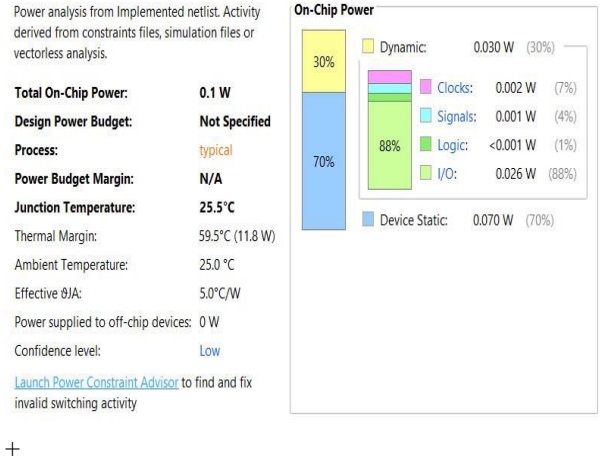

Fig. 6. On-Chip Power Analysis

The power analysis of the implemented Radix-4 FFT architecture shows a total on-chip power consumption of approximately 0.1 W. The design demonstrates efficient power usage, with minimal logic power contribution due to optimized architecture.

A significant portion of dynamic power is consumed by I/O operations, while static power dominates overall consumption. The results confirm that the proposed design is suitable for low-power and high-performance signal processing applications.

VII. Comparison with Existing Work

The proposed Radix-4 FFT architecture is compared with conventional Radix-2 FFT designs as well as other optimized FFT implementations reported in literature. The comparison is based on key performance metrics such as latency, throughput, memory usage, and hardware complexity.

Radix-2 FFT architectures are widely used due to their simplicity and ease of implementation. However, they require a larger number of computational stages, which increases latency and reduces overall efficiency. In contrast, the Radix- 4 FFT processes four inputs simultaneously, reducing the number of stages and improving performance.

Compared to existing pipelined FFT designs, the proposed architecture offers improved throughput due to its efficient pipeline structure and reduced stage count. Memory usage is also optimized through data reuse techniques, which minimize redundant storage and access operations.

While some advanced FFT architectures achieve high performance through complex designs, they often incur higher hardware costs and increased power consumption. The proposed design strikes a balance between performance and com plexity, providing significant improvements without excessive resource usage.

VIII. Conclusion

In this work, a high-throughput and memory-efficient Radix-4 FFT architecture has been successfully designed and implemented using Verilog HDL. The proposed architecture addresses the limitations of conventional FFT designs by reducing the number of computational stages and incorporating pipelining techniques to enhance throughput.

The use of Radix-4 decomposition significantly reduces latency, while the pipeline architecture enables continuous data processing. Memory optimization techniques such as data reuse and efficient register utilization further improve performance and reduce hardware complexity.

The design has been verified through simulation and synthesis using Xilinx Vivado, and the results demonstrate its effectiveness in terms of speed, efficiency, and resource utilization. The proposed architecture is well suited for realtime signal processing applications, including wireless communication systems, radar signal processing, and multimedia applications.

IX. Future scope

The proposed Radix-4 FFT architecture can be further enhanced in several ways to meet the growing demands of modern signal processing systems. One potential extension is the implementation of larger FFT sizes, such as 64-point, 256-point, or 1024-point FFTs, which are commonly used in communication systems. This would require scalable architecture design and efficient memory management techniques. Another area of improvement is the development of multi-channel FFT architectures, which can process multiple input streams simultaneously. This is particularly useful in applications such as MIMO communication systems and radar signal processing.

Integration with Orthogonal Frequency Division Multiplexing (OFDM) systems is another promising direction. Since FFT is a fundamental component of OFDM, the proposed architecture can be adapted for use in wireless communication standards such as 4G and 5G.

Power optimization techniques, such as clock gating and dynamic voltage scaling, can be applied to further reduce energy consumption. Additionally, the design can be extended for ASIC implementation, enabling its use in commercial products and high-performance DSP processors Future work includes:

- Implementation of larger FFT sizes
- Multi-channel FFT processing
- Integration with OFDM communication systems
- Power optimization techniques for low-power Applications

X. References

[1] T. Hu, C. Hao, X. Wang, Z. Liu, S. Ren, Z. Xu, and S. Zhu, "Efficient FPGA implementation of multi-channel pipelined large FFT architectures based on SA-MDF algorithm," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 72, no. 5, pp. 2188–2202, May 2025.

[2] P. Surya, C. Arunachalaperumal, and S. Dhilipkumar, "Performance estimation of low-power and area-efficient parallel pipelined FFT," 2025.

[3] P. Paz and M. Garrido, "A 12.8-GS/s 32-parallel 1 millionpoint FFT," in *Proc. 39th Conf. Design Circuits Integer. Syst. (DCIS)*, Nov. 2024, pp. 1–6.

[4] C. Yang, J. Wu, S. Xiang, L. Liang, and L. Geng, "A highthroughput and flexible architecture based on a reconfigurable mixed-radix FFT with twiddle factor compression and conflictfree access," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 31, no. 10, pp. 1472–1485, Oct. 2023.

[5] X. Zhou, X. Chen, Y. He, and X. Mou, "A flexible-channel MDF architecture for pipelined radix-2 FFT," IEEE Access, vol. 11, pp. 38023–38033, 2023.

[6] M. Garrido, "A survey on pipelined FFT hardware architectures," *J. Signal Process. Syst.*, vol. 94, no. 11, pp. 1345–1364, Nov. 2022.

[7] N. K. Unnikrishnan and K. K. Parhi, "Multi-channel FFT architectures designed via folding and interleaving," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, 2022, pp. 142–146.

[8] Y. Li, H. Chen, and Y. Xie, "An FPGA-based four-channel 128K-point FFT processor suitable for spaceborne SAR," *Electronics*, vol. 10, no. 7, p. 816, Mar. 2021.

[9] M. Garrido and P. Malagon, "The constant multiplier FFT," IEEE Trans.Circuits Syst. I, Reg. Papers, vol. 68, no. 1, pp. 322– 335, Jan. 2020.

[10] J. Wang, S. Li, and X. Li, "Scheduling of data access for the radix-2k FFT processor using single-port memory," IEEE Trans. Very Large Scale Integer. (VLSI) Syst., vol. 28, no. 7, pp. 1676– 1689, Jul. 2020.

[11] H. Mahdavi and S. Timarchi, "Area-time-power efficient FFT architectures based on binary-signed-digit CORDIC," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 66, no. 10, pp. 3874–3881, Oct. 2019.

[12] H. Kanders, T. Mellqvist, M. Garrido, K. Palmkvist, and O. Gustafsson, "A 1 million-point FFT on a single FPGA," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 66, no. 10, pp. 3863–3873, Oct. 2019.

[13] M. Garrido, K. Möller, and M. Kumm, "World's fastest FFT architectures: Breaking the barrier of 100 GS/s," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 66, no. 4, pp. 1507– 1516, Apr. 2019.

[14] J. G. Nash, "Distributed-memory-based FFT architecture and FPGA implementations," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 65, no. 9, pp. 1314–1318, Sep. 2018.

[15] E. Fankhauser and J. Wassner, "FPGA implementation of a multichannel continuous-throughput FFT processor," in *Proc. IEEE Int. Workshop Signal Process. Syst. (SiPS)*, 2018, pp. 181– 186.

[16] X.-Y. Shih, H.-R. Chou, and Y.-Q. Liu, "Design and implementation of flexible and reconfigurable SDF-based FFT chip architecture with changeable-Radix processing elements," IEEE Trans. Circuits Syst. I,Reg. Papers, vol. 65, no. 11, pp. 3942–3955, Nov. 2018.

[17] T. Liu, "Mapping Large 2D FFTs onto FPGAs using high level synthesis,"Master's thesis, Dept. Math. Comput. Sci. Archit. Inf. Syst. Res.Group, Eindhoven Univ. Technol., Eindhoven, The Netherlands, 2016.

[18] F. Qureshi and O. Gustafsson, "Generation of all radix-2 fast Fourier transform algorithms using binary trees," in Proc. 20th Eur. Conf. Circuit Theory Design (ECCTD), Aug. 2011, pp. 677– 680.

[19] T.-Y. Sun and Y.-H. Yu, "Memory usage reduction method for FFT implementations on DSP based embedded system," in Proc. IEEE 13th Int. Symp. Consume. Electron., May 2009, pp. 812–815.

[20] C. He, W. Qiang, G. Zhenbin, and W. Hongxing, "A pipelined memory efficient architecture for ultra-long variablesize FFT processors," in Proc. Int. Conf. Comput. Sci. Inf. Technol., Sep. 2008, pp. 357–361.

[21] H.-Y. Lee and I.-C. Park, "Balanced binary-tree decomposition for area efficient pipelined FFT processing," IEEE Trans. Circuits Syst. I, Reg.Papers, vol. 54, no. 4, pp. 889– 900, Apr. 2007.

[22] B. A. Cipra, "The best of the 20th century: Editors name top 10 algorithms," *SIAM News*, vol. 33, no. 4, pp. 1–2, 2000

[23] E. O. Brigham, *The Fast Fourier Transform and Its Applications*. Upper Saddle River, NJ, USA: Prentice-Hall, 1988.

[24] W. T. Cochran, J. W. Cooley, D. L. Favin, H. D. Helms, R. A. Kaenel, W. W. Lang, G. C. Maling Jr., D. E. Nelson, C. M. Rader, and P. D. Welch, "What is the fast Fourier transform?," *Proc. IEEE*, vol. 55, no. 10, pp. 1664–1674, Oct. 1967.

[25] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series," *Math. Comput.*, vol. 19, no. 90, pp. 297–301, 1965.