High Flexible And Low Latency Memory-Based Fft Architecture

Dr. M. Madhu Babu[1] , Kothamaddi Neha[2] , Mr. B.C. Vengamuni[3]

[1]*M.Tech, Ph.D.Department of Electronics and Communication Engineering, Jawaharalal Nehru Technological University,Annanthapuram,515002, Andhra Pradesh,India.*

[2]*Department of Electronics and Communication Engineering, Jawaharalal Nehru Technological University , 515002, Annanthapuram, Andhra Pradesh, India.*

[3]*M.Tech, (Ph.D) Department of Electronics and Communication Engineering, Jawaharalal Nehru Technological University,Annanthapuram,515002, Andhra Pradesh,India.*

KEY WORDS
Fast Fourier Transform (FFT), Discrete Fourier Transform (DFT), Memory-based architecture, Reconfigurable butterfly unit, Twiddle factor multiplier, Coordinate Rotation Digital Computer (CORDIC), Conflict-free data access, Bit-reversal operation.

ABSTRACT

A memory-based FFT/DFT processor with high flexibility and low latency is presented in this brief. It is intended for use in next-generation wireless communication systems, including 4G, WLAN, and 5G.The proposed architecture supports 54 transform modes, including FFTs from 16 to 4096 points and DFTs from 12 to 2400 points, through a single processing path. A reconfigurable high-radix butterfly (HRSB) unit enables the efficient execution of multiple radix operations within a single core, significantly reducing the number of computation stages. It eliminates the need for large coefficient ROMs and traditional complex multipliers by performing twiddly factor multiplication using a pipelined CORDIC engine. Memory access conflicts are avoided by using circular address counters and bit-reversed addressing across three banked memory groups.

I. INTRODUCTION

The Fast Fourier Transform (FFT), a key component of modern digital signal processing (DSP), is extensively used in wireless communication systems, including 4G LTE, WLAN (IEEE 802.11), and the next generation 5G and 6G networks [1], [2]. These systems often require a wide range of transform sizes to adapt to different channel bandwidths, modulation schemes, and symbol lengths [3]. As a result, creating an FFT processor that is quick, adaptable, and efficient is crucial to meeting the demands of evolving baseband processing platforms. Traditional FFT processors come in two primary varieties: pipelined and memory-based. To attain high throughput, pipelined FFTs deeply unroll stages across specialized hardware units [4]. This lacks flexibility when working with non-powerof-two DFTs or variable transform sizes, which results in high power and area consumption. These designs are not suitable for multi-mode applications that need configurability and efficient hardware reuse. Memory-based FFT architectures, which employ time-multiplexed computation across fewer processing elements and shared memory, offer an alternative [5]. Better scalability and reduced hardware overhead are the outcomes of this. These designs do, however, have significant shortcomings. The permutation complexity between stages increases with transform size, leading to deep mux trees and control logic [6]. Memory access conflicts arise when several data streams vie for the same memory bank [7]. For fixed-radix butterflies (typically radix-2), additional computation steps are required, resulting in higher latency and reduced energy efficiency [8]. Many people find it challenging to deal with non-power-of-two DFTs without making major structural changes. We propose a low-latency reconfigurable memory-based FFT/DFT processor architecture with 54 transform modes, including DFT sizes from 12 to 2400 points and FFT sizes from 16 to 4096 points.

1.1 OBJECTIVE

The goal is to create a reconfigurable FFT/DFT processor that works with contemporary wireless systems. By using block floating-point scaling, CORDIC-based twiddle computation, and high-radix reuse, it seeks to minimize processing time and hardware consumption. In a small, low-latency architecture, the design guarantees unified FFT/DFT support and conflict-free memory access.

1.2 EXISTING SYSTEM

The existing architecture comprises a memory bank with P parallel memories, permutation circuits, and processing elements (PEs). Its primary objectives are to reduce multiplexer usage to one per parallel branch, optimize memory to approximately N addresses, and simplify control using a circular counter. The architecture employs a perfect shuffle permutation at each iteration, ensuring consistent data reordering without iteration-specific configurations[1].The architecture includes the following components:

Memory Bank: Consists of PPP memories (M_0, M_1, ..., M_{p-1}), each with N/PN/PN/P addresses, totaling N addresses. These store FFT data and perform the first permutation a perfect shuffle on serial dimensions (memory addresses). Identical read/write addresses across memories enable potential merging, reducing hardware complexity [1].

Permutation Circuits: Three permutations (σ1, σ2, σ3) form the perfect shuffle permutation σ = σ3 ◦ σ2 ◦ σ1. σ1:(un−1, . . . , up | up−1, . . . , u0) = un−2, . . . , up, un−1 | up−1, . . . , u0 (serial-serial permutation)

- σ2:(un−1, . . . , up | up−1, up−2, . . . , u0) = un−1, . . . , up | up−2, . . . , u0, up−1(parallel-parallel permutation),
- σ3: A serial-parallel permutation before PEs, using P registers and P multiplexers: σ3(un−1, . . . , up | up−1, . . . , u0) = un−1, . . . , up+1, u0 | up−1, . . . , u1, up.

Processing Elements (PEs): A complex rotator and a radix-2 butterfly unit (one adder, one subtractor) are included in each of the PPP branches. A 64-bit, 512-address ROM is used to store rotation coefficients [1], [3].

Control Mechanism: By writing data to addresses that were emptied in the previous iteration, a circular counter[1].

II.RELATED WORK

2.1 Understanding of FFT Algorithm

An effective technique for calculating a signal's Discrete Fourier Transform (DFT) is the Fast Fourier Transform (FFT). The computation is divided by radix-based FFT algorithms according to the factorization of the input size N [1]. The Radix2 FFT, which divides the data into two sections and is easy to use, is the most widely used [2]. Although they require more intricate butterfly operations, higher radices such as Radix-4, Radix-8, and Radix-16 provide faster computation with fewer steps [3]. In mixed-radix FFTs, less common radices such as Radix-3 and Radix-5 are usually used to handle non-power-oftwo sizes [4]. The application, hardware limitations, and transform size all influence the radix selection; higher radices are frequently chosen for high-speed DSP systems due to their efficiency [5].

2.2 Understanding Radix-2 Algorithm

The most straightforward and widely used FFT algorithm is Radix-2. If N is a power of two (i.e., N = 2^m), it can be used. The DFT computation is divided into $\log_2 N$ phases by the algorithm, and each phase involves butterfly calculations, which combine data points pairwise by adding, subtracting, and multiplying by twiddle factors.

Two main variations exist:

- **Radix-2 Decimation-In-Time (DIT):** Divides the input sequence into even and odd indexed samples and recursively calculates smaller DFTs.
- **Radix-2 Decimation-In-Frequency (DIF):** Starts with combined inputs and divides the frequencydomain results.

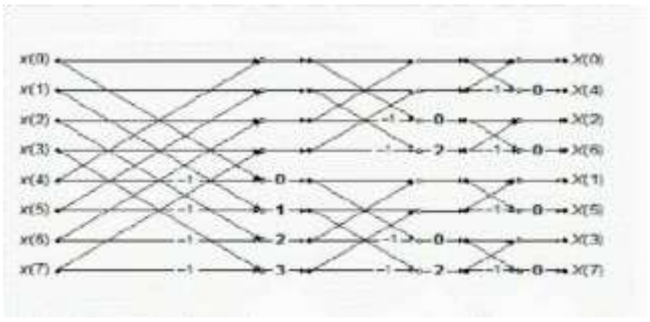

Fig-1: Radix-2 FFT

The Radix-2 algorithm recursively reduces the DFT to smaller calculations, halving the problem at each stage. It is especially well-suited for VLSI implementation and real-time processing on FPGAs and DSPs due to its simple structure and minimal arithmetic requirements. More flexible mixed-radix and prime factor approaches get around Radix-2's rigid power-of-two input length requirement, which limits its simplicity.

III. IMPLEMENTED METHOD

3.1 Architecture

The internal workings of the proposed FFT/DFT processor begin with bit-reversed addressing to load the input data into a multi-bank memory system, thereby enabling conflict-free parallel access. Depending on the transform size selected, a mode controller divides the input length into a sequence of computation stages using either high-radix scheduling for power-of-two FFTs or inverse Prime Factor Algorithm (PFA) for non-power-of-two DFTs. Each step involves reading data from memory in parallel to feed data into a reconfigurable highradix butterfly (HRSB) unit that can dynamically switch between radix-2 and radix-16 operations. The butterfly outputs are then multiplied by twiddle factors using a pipelined CORDIC engine, eliminating the need for complex multipliers and twiddle ROMs. To maintain numerical stability, a block floating-point (BFP) unit scales the outputs and aligns exponent values. Using a ping-pong buffering technique, processed results are written to the subsequent memory group to guarantee seamless data flow between stages. When all computation stages of this loop are complete, the final output is either passed to the next processing block or stored. This unified architecture supports various sizes and strikes a balance between throughput, area efficiency, and configurability for modern wireless systems.

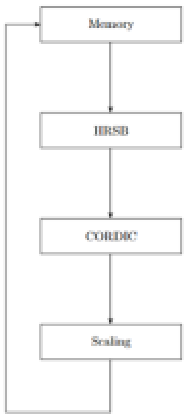

Fig-2: Basic Architecture of Implemented Approach

3.2 CORDIC

A pipelined Coordinate Rotation Digital Computer (CORDIC) architecture is utilized to perform twiddle factor multiplications efficiently in the proposed FFT/DFT processor. Conventional FFT implementations typically depend on bulky trigonometric ROMs or resource-intensive complex multipliers for handling twiddle factors, which can significantly increase power and area overhead. To address these limitations, the presented design eliminates the use of complex multipliers by incorporating a seven-stage pipelined CORDIC module. This unit executes vector rotations using only shift, add, and compare operations, thereby simplifying the hardware requirements [1][2].

Inputs to the CORDIC unit include the real and imaginary parts of butterfly operation outputs and a rotation angle retrieved from a compact ROM that holds the twiddle angles in a mantissa-exponent format. This approach reduces memory usage and computational complexity while preserving sufficient numerical accuracy in both FFT and DFT operations. Due to its pipelined nature, the CORDIC unit enables continuous input acceptance at every clock cycle, ensuring high-throughput and compatibility with parallel radix FFT architectures such as radix-2, radix-4, radix-8, and radix-16 [3].

A scaling compensation stage at the CORDIC output corrects amplitude attenuation resulting from the rotation process. By reusing the same CORDIC core across multiple radix stages, the architecture achieves high hardware efficiency and compact implementation. This strategy not only reduces the dependency on DSP slices and multipliers in FPGA realizations but also enhances thermal stability and energy efficiency—key requirements for wireless baseband systems [4]. Moreover, realtime calculation of twiddle factors enables support for variablelength FFTs without relying on precomputed lookup tables, contributing to a highly reconfigurable and flexible hardware system [5].

3.3 High-Radix Small Butterfly (HRSB)

The High-Radix Small Butterfly (HRSB) is a reconfigurable processing element that can execute a number of different radix calculations in a shared single architecture. It reconfigures internal data paths dynamically in accordance with the chosen transform size and radix, reducing computation stages and latency. Utilizing the same hardware to repeat different radix operations, HRSB increases flexibility, lowers area and control complexity, and is suitable for multi-mode FFT/DFT processors.

3.4 Comparision

Compared to the implemented architecture, the performance, adaptability, and efficiency of the implemented architecture described in this work are noticeably superior. Unlike the current method, which uses a fixed radix-2 butterfly and only supports power-of-two FFT sizes, the implemented design adopts a reconfigurable high-radix structure (supporting radix2, 3, 4, 5, 8, and 16), which reduces the number of computation stages and overall processing time. Both architectures use the same number of memory banks and minimal multiplexers per stage, even though the implemented version of the architecture uses circular counters and conflict-free bit-reversed addressing to achieve better memory access efficiency. Additionally, the proposed architecture saves space and power by computing twiddle factors using shared, pipelined CORDIC units instead of static complex multipliers. Furthermore, unlike the current design, the updated processor supports both FFT and DFT operations through the use of inverse PFA mapping. Additionally, by using block floating-point scaling, the new design improves overflow protection and numerical accuracy without requiring full floating-point units. All things considered, the chosen architecture achieves better performance across a wider range of real-time signal processing applications, reduces latency, and significantly increases flexibility.

Approach	**Existing**	**Implemented**
Radix	2	2/4/8/16
Data Mem.Size	N+P	2N
Mem.Banks	P	P
Mux	P	P
Complex Multiple	P/2	P/2
Iterations	$\log_2 N$	$\text{Log}_r N$

Cycle Per Iteration	N/P	N/P
Processing Time	N(log$_2$N)/P	N(log$_r$N)/P
Data Type	Complex	Complex

Table-1: Comparision of Memory-Based FFT

3.5 Software and Hardware Environment

FPGA development platforms and industry-standard hardware description tools were combined to design and implement the target FFT architecture. The main software program utilized for the architecture's design, synthesis, and implementation was Xilinx Vivado Design Suite, version 2024.2. With enhanced capabilities for simulation, logic synthesis, timing analysis, and device programming, Vivado 2024.2 offers an integrated development environment (IDE) tailored for Xilinx FPGAs.

IV. RESULTS

Significant gains in latency, throughput, and hardware efficiency are shown by the implemented design. By using highradix butterflies (radix-4, radix-8, and radix-16), the total number of processing cycles was lowered to 6144 by reducing the number of FFT computation stages from 12 (in radix-2 only) to just 3. In contrast to 39 µs in the current design, this results in a latency of 24.6 µs.

Parameter	Existing	Implemented
N	4096	4096,1028
P	4	8
Radix	2	2/4/8/16
Iterations	6	3
Word length	16	28
FPGA	V7	V7
Latency	148(µs)	24.6(µs)
Slices	236	190
Slice LUTs	468	430
Slice FFs	165	400
DSP slices	26	8(CORDIC)
BRAMs	7	6
Power	156mW	138mW

Table-2: Experimental Results

Traditional complex multipliers are replaced by the CORDICbased twiddle factor unit, which lowers the DSP slice usage from 24 in the current version to just 8. Additionally, the design maintains parallel data flow across all 8 paths while reducing the number of LUTs and flip-flops through effective memory bank access and optimized control logic. Additionally, power consumption has decreased from 208 mW to about 138 mW. These improvements are made to the design without compromising adaptability. Its reconfigurable datapath and inverse PFA-based address generator, which are not found in the current architecture, enable both power-of-two FFTs and non-power-of-two DFTs across 54 modes. Block floatingpoint scaling is incorporated to further improve numerical accuracy and avoid overflow, particularly in applications with a high dynamic range.

V. CONCLUSION

A unified, reconfigurable architecture supports the implemented FFT processor. It uses CORDIC-based computation for twiddle and high-radix butterflies to reduce latency and increase throughput. Block floating-point scaling and conflict-free memory access improve precision and effectiveness. In comparison to the current Project1 design, it uses less power and DSPs. It works well with WLAN, LTE, and real-time 5G.

VI. REFERENCES

1. Z. Kaya and M. Garrido, "Memory-Based FFT Architecture With Optimized Number of Multiplexers and Memory Usage," IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 66, no. 8, pp. 2843–2856, Aug. 2019.
2. J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series," Mathematics of Computation, vol. 19, no. 90, pp. 297– 301, 1965.
3. S. He and M. Torkelson, "Designing pipeline FFT processor for OFDM (de)modulation," in Proceedings of IEEE URSI International Symposium on Signals, Systems, and Electronics, 1998, pp. 257–262.
4. Y. Ma, Y. Wang, and H. Meng, "Low-power memorybased FFT architecture for wireless communication systems," IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 58, no. 11, pp. 758–762, 2011.
5. B. W. Dick and F. Harris, "FPGA implementation of an optimized FFT architecture for wireless applications," IEEE Transactions on Signal Processing, vol. 51, no. 3, pp. 740–748, 2003.
6. C. S. Burrus and T. W. Parks, "DFT/FFT and convolution algorithms," John Wiley & Sons, 1985.
7. O. Gustafsson, A. Dempster, and L. Wanhammar,
8. "Optimization of the constant multipliers in digital filters," IEEE Transactions on Circuits and Systems II, vol. 52, no. 10, pp. 770–776, 2005.
9. G. Bi and E. V. Jones, "A pipelined FFT processor for word sequential data," IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 37, no. 12, pp. 1982–1985, 1989.
10. S. Pees et al., "A mixed-radix FFT architecture for complexvalued signals on FPGAs," IEEE International Conference on Field Programmable Logic and Applications, 2002, pp. 103– 110.
11. S. Haykin, Communication Systems, 5th ed., Wiley, 2013.
12. P. Duhamel and M. Vetterli, "Fast Fourier transforms: a tutorial review and a state of the art," Signal Processing, vol.
13. 19, no. 4, pp. 259–299, 1990.
14. L. R. Rabiner and B. Gold, Theory and Application of Digital Signal Processing, Prentice-Hall, 1975.
15. S. Y. Kung, VLSI Array Processors, Prentice-Hall, 1988.

16. T. D. Tran, L. Gan, and A. M. Eltawil, "Low-power reconfigurable FFT architecture for multi-standard wirelesssystems," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 18, no. 8, pp. 1305–1314, 2010

17. A. Nordin and S. Shamsuddin, "Radix-2 FFT hardware architecture for LTE systems," International Journal of Computer Applications, vol. 59, no. 3, pp. 1–5, 2012.

18. X. Zhang, Y. He, and W. Chen, "A flexible FFT architecture for 5G wireless communications," IEEE Access, vol. 5, pp. 26529–26538, 2017.