High-Throughput Reconfigurable FFT Processor with Approximate Block Floating-Point Arithmetic and Conflict-Free Memory Access

[1]Bynagari Abraham Ron Clayton, [2] Dr.J.Amarendra, [3]V.Bhaskar Rao

[1]M.Tech Scholar, Dept of ECE, Audisankara College of Engineering and Technology, Gudur, Andhra Pradesh.

[2]Professor&Principal, Dept of ECE, Audisankara College of Engineering and Technology, Gudur, Andhra Pradesh.

[3]Assistant professor, Dept of ECE, Audisankara College of Engineering and Technology, Gudur, Andhra Pradesh.

[1]ronclayton538@gmail.com , [2]amarendra.ece@audisankara.ac.in , [3]bhaskarrao.vennam@gmail.com

ABSTRACT: This paper proposes a novel Fast Fourier Transform hardware architecture for natural-order multi-input multi-output processing, which is based on approximate computing and the block floating-point data format. The architecture innovatively combines the low-power characteristics of approximate multipliers with the high-precision advantages of block floating-point data formats, enabling efficient processing of 2048-point and 4096-point Fast Fourier Transform data. By employing conflict-free memory access techniques, the design optimizes data scheduling, reduces idle cycles during processing, and significantly enhances energy efficiency while maintaining computational accuracy. Experimental validation demonstrates that the processor, fabricated using a 40nm process, achieves a compact chip area of 0.42 mm2 . Operating at a frequency of 671.14 MHz, it delivers a throughput of 1073.82 MS/s for 4096-point FFT computations, with a power consumption of only 110.85 mW and a Signal–to–Quantization Noise Ratio of 58.51 dB. Compared to state-of-the-art designs, this architecture achieves breakthrough improvements in throughput performance while maintaining competitive energy efficiency.

INDEX TERMS Approximate computing, block floating-point, conflict-free access, memory-based fast Fourier transform FFT, mixed radix.

I. INTRODUCTION

The Fast Fourier Transform (FFT), as a fundamental algorithm in digital signal processing (DSP) and data analysis, has been widely employed in numerous applications ranging from wireless communication systems to medical imaging, audio processing, and radar technologies [1], [2], [3]. There exist numerous hardware architectures for realizing an FFT processor and each offering distinct advantages. Firstly, the pipelined architecture [4], [5], [6], which processes data one sample at a time, is characterized by independent butterfly units and dedicated memory at each stage, enabling continuous high-speed data processing and improved throughput. Instead, the goal of memory-based FFT architecture [7], [8], [9], [10], [11], [12] is to reduce the area and hardware resources usage. The reuse of the butterfly units for

different stages reduces the number of processing elements and, therefore, the area of the circuit. The architecture provides a more compact design by reusing the butterfly units and memories. For small-size FFT processing, the hardware implementation complexity is low, and pipeline design is more commonly adopted. As the size increases, hardware design complexity rises, and memory-based FFT architectures become the preferred choice. FFT processors are computationally intensive and memory-intensive architectures that require substantial computational and storage resources. In certain applications, such as audio processing and image compression, users have relatively low precision requirements for FFT results and can tolerate a certain degree of error. Approximate computing reduces the precision demands of operations, thereby decreasing hardware resource usage and lowering dynamic power consumption. Yan et al. [14] present a top-down design strategy for approximate floating-point FFTs, utilizing a mantissa bit-width adjustment algorithm and a step by step multiplier approximation algorithm, achieving 50% area reduction and 70% power-delay product reduction compared to a design with 60 dB Signal Noise Ratio (SNR). To further increase FFT performance, Reddy and Kumar [15] introduces an approximate radix-8 multiplier to replace exact multipliers, resulting in 20% performance improvement over traditional FFTs which use radix-8 multipliers. Du et al. [16] propose four different levels of approximate radix-4 multipliers, achieving a 69.9% reduction in power consumption and the delay is reduced up to 45.7% within pipelined FFT implementations. Pereira et al. [17] explores optimal design combinations at various levels of approximation through the use of approximate multipliers and adders, achieving a minimum decrease of 20.2% in power consumption without degrading the quality of the spectrum generation. Santana et al. [18] enhances the power efficiency of butterfly computing units by using an efficient 5-2 adder compressor, resulting in a power-efficiency improvement of up to 47.28%. Ferreira et al. [19] explore the precision loss of different approximation levels by substituting a set of approximate adders for precise adders in the butterfly, and the best-proposed FFT achieves a 35% reduction in power consumption. Liu et al. [20] proposes two algorithms for modifying word lengths under specified error margin, allowing for a trade-off between precision and performance, with a 27% power reduction and a precision loss of less than 2%. With the advancement of edge computing, the Internet of Things (IoT), and neural networks, approximate FFT designs are poised to play a significant role in a broader range of application scenarios. In this paper, we propose a new approximate FFT processor based on combining approximate multiplication with block floating-point algorithm, which aims to reduce hardware cost through approximate computation and maintain high computation accuracy and efficiency. By carefully designing the memory accesses to minimize conflicts during FFT computation. By aligning with the memory addressing patterns and the data dependencies in the FFT algorithm, we are able to reduce the number of idle cycles, resulting in a more efficient use of the

available memory bandwidth and processing resources, thereby enhancing the efficiency and throughput of the overall implementation.

II. LITERATURE REVIEW:

Memory-based FFT architecture with optimized number of multiplexers and memory usage

This brief presents a new -parallel radix-2 memory-based fast Fourier transform (FFT) architecture. The aim of this brief is to reduce the number of multiplexers and achieve an efficient memory usage. One advantage of the proposed architecture is that it only needs permutation circuits after the memories, which reduces the multiplexer usage to only one multiplexer per parallel branch. Another advantage is that the architecture calculates the same permutation based on the perfect shuffle at each iteration. Thus, the shuffling circuits do not need to be configured for different iterations. In fact, all the memories require the same read and write addresses, which simplifies the control even further and allows to merge the memories. Along with the hardware efficiency, conflict-free memory access is fulfilled by a circular counter. The FFT has been implemented on a field programmable gate array. Compared to previous approaches, the proposed architecture has the least number of multiplexers and achieves very low area usage.

A 4096- point radix-4 memory-based FFT using DSP slices,'

This brief presents a novel 4096-point radix-4 memory-based fast Fourier transform (FFT). The proposed architecture follows a conflict-free strategy that only requires a total memory of size N and a few additional multiplexers. The control is also simple, as it is generated directly from the bits of a counter. Apart from the low complexity, the FFT has been implemented on a Virtex-5 field programmable gate array (FPGA) using DSP slices. The goal has been to reduce the use of distributed logic, which is scarce in the target FPGA. With this purpose, most of the hardware has been implemented in DSP48E. As a result, the proposed FPGA is efficient in terms of hardware resources, as is shown by the experimental results.

Design of high hardware efficiency approximate floating-point FFT processor,"

The Fast Fourier Transformation (FFT), as a high-efficiency algorithm of the Discrete Fourier Transform (DFT), is widely used in Digital Signal Processing (DSP), wireless communication systems, spectrum analysis, and image processing. Approximate computing has shown effectiveness and feasibility to enhance the hardware efficiency of these applications. However, most approximate units in previous works are designed case by case, which has low efficiency and is difficult to find the optimal design. In this paper, a top-down design strategy for approximate floating-point (FP) FFT is proposed, which includes a mantissa bit-width adjustment algorithm and a step-by-step multiplier approximation algorithm. With the mantissa bit-width adjustment algorithm, the approximate 64 FP FFT achieved 50% area reduction and 70% power-delay product (PDP) reduction compared to the exact design with a 60dB Signal Noise Ratio (SNR) requirement,

which is also at least 52% and 33% better than the previous approximate FP FFT. After using the step-by-step multiplier approximation algorithm, the approximate mantissa multiplier with an 8-bit fractional part reduced the area and PDP by 81.15% and 93.70%, respectively. The feasibility of the proposed approximate FFT design is verified in the channel estimation module of a wireless communication system, spectrum analysis, and image processing system.

‘‘Performance analysis of 8-point FFT using approximate radix-8 booth multiplier

Most commonly used algorithms in digital signal processing are Fast Fourier Transform (FFT) and Inverse Fast Fourier Transform (IFFT). In FFT butterfly unit is the basic unit which contains components like multipliers and adders. Implementing these components plays a very tough role in designing of butterfly structure. To increase the performance of FFT further an approximate radix-8 Booth multiplier using approximate full recoding adder and Kogge-Stone adder is proposed. By using proposed multiplier performance of the FFT is increased by 20 %. When compared with conventional FFT using normal radix-8 booth multiplier.

Design of an approximate FFT processor based on approximate complex multipliers

The Fast Fourier Transform (FFT) is an efficient algorithm to calculate the Discrete Fourier Transform (DFT), which is often employed in Digital Signal Processing (DSP) and communication. In FFT, complex multiplication and addition units in the butterfly module consume most of the hardware resources. Compared to the addition operation, multiplication is more complicated. In this paper, the multiplier in the complex multiplication unit of the FFT is approximated. Four Radix-4 Booth multipliers with different approximation levels are proposed to reduce the hardware complexity. The pipeline FFT and the parallel FFT based on the proposed approximate multipliers are implemented and extensively evaluated. Compared with the state-of-the-art FFT designs, the LUTs amount is reduced up to 20.3% and 29.1% for pipeline and parallel FFTs, respectively. The power is reduced up to 69.9% for pipeline FFT, and the delay is reduced up to 45.7%. Moreover, the PSNR is reduced by less than 1dB in both pipeline FFT and parallel FFT. Proved by experiment results, the overall performance of the proposed designs is better than FFT designs using other approximate multipliers.

Energy-quality scalable design space exploration of approximate FFT hardware architectures,

This paper presents a comprehensive design space exploration for boosting energy efficiency of a fast Fourier transform (FFT) VLSI accelerator, exploiting several approximate multipliers (AxM) combined with approximate adder (AxA) circuits. The FFT hardware herein presented consists of a fixed-point sequential architecture using a radix-2 butterfly with decimation in time. We explore a set of AxMs – namely Dynamic Range Unbiased (DRUM), Rounding-based Approximate (RoBA), leading one Bit-based Approximate (LoBA), and Truncated approach – jointly with the LOA, ETA-I, CopyA, CopyB, Trunc0, Trunc1 approximate adders. The approximate arithmetic operators

are used in the butterfly kernel with exploration of the approximation levels (for the and least-significant bits, respectively, for the AxM and AxA), aiming at discovering the most energy-efficient configuration under a design-time QoR constraint. The mean square error and peak signal-to-noise ratio metrics define which approximate levels combining and variations will enable the FFT to process signals to generate spectrograms without significant losses. Our results show that the LoBA multiplier with =8 together with the LOA, Trunc1 and Trunc0, at different approximation levels, provide most energy savings with controllable quality degradation, presenting a minimum decrease of 20.2% in power dissipation without degrading the spectrogram generation quality.

Using efficient adder compressors with a split-radix butterfly hardware architecture for low-power IoT smart sensors

Fast Fourier Transform (FFT) is one of the most common implementations of the Discrete Fourier Transform (DFT), and it is a commonly used algorithm to process and classify data in IoT (Internet of Things) smart sensors. Butterflies play a central role in FFT computation since they allow calculation of complex terms. In this work, we propose a power-efficient hardware architecture for 16-bit split-radix DIT (Decimation in Time) butterflies. Based on the results we show that the 16-bit split-radix butterfly hardware architecture is more power-efficient than 16-bit radix-4 one. Moreover, we are able to improve the power-efficiency of the split-radix butterfly by using efficient 5-2 adder compressors. The results show that our best proposed split-radix saves up to 47.28% of power dissipation by using 5-2 adder compressors, when compared with radix-4 butterfly using the synthesis tool adder.

A powerefficient FFT hardware architecture exploiting approximate adders,'

This work presents an energy-efficient Fast Fourier Transform (FFT) hardware architecture exploiting approximate adder circuits. The FFT hardware architecture consists of a fixed-point fully sequential architecture with a radix-2 butterfly with decimation in time (DIT). In this paper, we explore a set of approximate adders (LOA, ETA-I, Copy-A, Copy-B, Trunc0, Trunc1) in the butterfly by varying the approximation level (K term). The Root-Mean-Square Error (RMSE) metric shows which approximate level term allows the FFT processing without widely signal losses. The results show that our best-proposed FFT employing Trunc0 approximate adder with K =10 saves up to 35% of power dissipation compared to the FFT with the original radix-2 butterfly using the synthesis tool operators.

Approximate designs for fast Fourier transform (FFT) with application to speech recognition

This paper presents different approximate designs for computing the FFT. The tradeoff between accuracy and performance is achieved by adjusting the word length in each computational stage. Two algorithms for word length modification under a specific error margin are proposed. The first algorithm targets an approximate FFT for an area-limited design compared to the conventional fixed design; the second

algorithm targets performance so it achieves a higher operating frequency. Both of the proposed algorithms show that an efficient balance between hardware utilization and performance is possible at stage-level. The proposed approximate FFT designs are implemented on FPGA; experimental results show that hardware utilization using the first approximate algorithm are reduced by at least nearly 40%. The second algorithm increases performance of the designs by over 20%. Fine granularity design is also investigated, where the FPGA resources for a 256-point FFT computation can be further reduced by nearly 10% compared to a coarse design. Finally, the proposed approximate designs are applied to a feature extraction module in an isolated word recognition system; the numbers of LUTs and FFs for the Mel frequency cepstrum coefficients (MFCC) extraction module are decreased by up to 47.2% and 39.0%, respectively with a power reduction of up to 27.0% at a loss in accuracy of less than 2%.

A memory-based FFT processor design with generalized efficient conflict-free address schemes

This paper presents the design and implementation of memory-based fast Fourier transform (FFT) processors with generalized efficient, conflict-free address schemes. We unified the conflict-free address schemes of three different FFT lengths, including the single-power points, the common nonsingle-power points, and the nonsingle-power points applied with a prime factor algorithm. Though the three cases differ in terms of decomposition, they are all compatible with memory-based architecture by the way of the proposed address schemes. Moreover, the decomposition algorithm utilizes a method, named high-radix-small-butterfly (HRSB), to decrease the computation cycles and eliminate the complexity of the processing engine. In addition, an efficient index generator, a simplified multipath delay commutator engine, and a unified Winograd Fourier transform algorithm butterfly core were also designed. We designed two FFT examples in long-term evolution system to verify the availability of the address scheme, including a 2n (128-2048)-point FFT unit and a 35 different point (12-1296) DFT unit. Compared with previous works with similar address schemes, this paper supports more generalized lengths and achieves more flexible throughput.

‘‘A modified signal flow graph and corresponding conflict-free strategy for memory-based FFT processor design

In this brief, we propose a novel method which realizes conflict-free strategy in memory-based FFT, of which the hardware complexity is simplified, since only a few extra registers are needed and the control logic is identical in all stages. In addition, we present a modified signal flow graph to fit for the proposed conflict-free strategy. The modified signal flow graph derives from the mixed-radix signal flow graph and has constant geometry property. Furthermore, continuous-flow is adopted to increase the throughput. Thus, the proposed FFT processor has better performance compared with the previous memory-based FFT processors. Simulation result shows that for the proposed 8 to 2048-point FFT processor,

the maximum frequency is 400MHz by using a 65-nm CMOS technology, and the area is 0.45 mm² in the same condition.

III. EXISTING METHOD

The solution is composed of an integrated FAST Fourier transform (FFT) is a compute-intensive algorithm in the physical layer of an orthogonal frequencydivision multiplexing (OFDM) system to convert data between time domain and frequency domain. Many OFDM systems such as 4G LTE/LTE-A [1] and wireless local area network (WLAN) [2], [3] require power-of-two FFTs. LTE uplink precoding requires nonpower-of-two discrete Fourier transforms (DFTs) from 12 to 2400. In the upcoming 5G [4] (the fifth generation mobile communication), FFT is still an essential algorithm for all of the waveform candidates, and the FFT computation speed should be high enough to support the high data rate of 5G. Therefore, in the future multimode base station, the FFT processor should support diverse DFTs and high-speed FFTs. Many high-speed FFT processors [5]–[13] have been proposed for power-of-two FFTs. However, there are a limited number of processors supporting nonpower-of-two DFTs. Processor in [14] adds an extra radix-3 unit to support the 1536-point DFT in 4G LTE. The single-path delay feedback (SDF) architecture in [15] supports 48 2m3n points using 6T-RC processing element and section-based twiddle factor (TF) generator (STFG). SDF processor in [16] supports 46 2m3n5k points using a single-table approximation method (STAM) for TF generation. However, the throughput is restricted to 1× clock rate because of the limitation of the single-path pipelined architecture. Processors in [17] and [18] support 128- to 2048-point FFTs and 12- to 1296-point DFTs and use the prime factor algorithm (PFA) to minimize the number of TF multiplication. However, the data access of PFA cannot be in parallel for data I/O (read in input and write out result). The throughput is thus restricted to 1× clock rate and cannot meet the high-speed requirement for future 5G. There are still challenges in designing a low-power high-speed yet flexible processor for DFTs and FFTs. To design a high-speed processor supporting DFTs and FFTs, several aspects should be considered, including: 1) butterfly unit(s) supporting 2-, 3-, 5-, and higher radices; 2) TF multiplication scheme with hardware efficiency, and 3) conflict-free data access scheme that supports multiple butterfly units for 2-, 3-, 5-, and higher radices as well as minimizes the memory usage for nonpower-oftwo DFTs. In this paper, the contributions are to propose a memory-based FFT processor supporting 54 modes including 16-4096 point FFTs and 12-2400 point DFTs for 4G, WLAN, and future 5G and propose improvements on critical parts in butterfly unit, TF multiplier, and data access scheme. By reusing the hardware resources under timing constraint, we propose a reconfigurable butterfly unit for eight radix-2 in parallel, four radix-3/4 in parallel, two radix-5/8 in parallel, and a radix-16 in one clock cycle. TF multipliers using different schemes are optimized to support both FFTs and DFTs. The proposed coordinate rotation digital computer (CORDIC) scheme is found to be the most hardware efficient scheme for this design. Furthermore, we add bit-reverse operation to the basic add-based data access scheme and support multiple butterfly units at any radix.

Compared with the state-of-the-art designs, the proposed processor supports the most kinds of transform sizes with the highest normalized throughput per area (Nor.Thrpt/area) (MS/s/mm2).

3.1.THE OVERALL ARCHITECTURE

The overall architecture of the FFT consists of three parts: computation, storage, and control. To ensure continuous and efficient data flow throughput the FFT computation process, a ping-pong buffer architecture is adopted. This structure comprises two groups of SRAM blocks, with each group containing 8 SRAM blocks. During data transfer, data codes are first read into storage unit A sequentially. The butterfly computation engine then receives the data to be processed from storage unit A and begins computation, after which the results are stored in storage unit B. Subsequently, data to be processed is retrieved from storage unit B and fed into the butterfly computation engine for the second stage of FIGURE 3. The overall architecture of FFT. computation. Upon completion, the results are stored back in storage unit A. This cycle continues, with data being output sequentially in the end to obtain the final results. The memory data bit-width is divided into two parts: the real part is stored in the higher-order bits, and the imaginary part in the lower-order bits. To optimize timing performance, registers are used at the output of the SRAM blocks to create a pipeline, which facilitates better synchronization with downstream processing units and alleviates potential timing violations. In this design, the input/output interface adopts an 8-channel parallel natural input/output scheme.

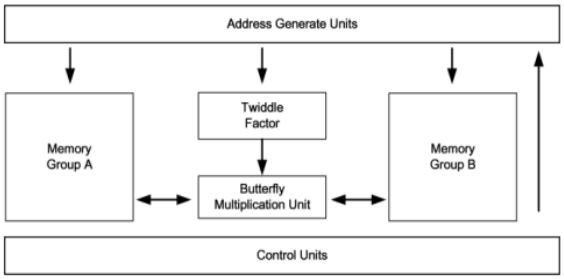


FIGURE 3. The overall architecture of FFT.

3.2.CONFLICT-FREE ACCESS

Conflict-free access technology is a method designed to minimize memory access conflicts by optimizing memory access patterns. In FFT algorithms, frequent data read/write operations make memory access conflicts a critical factor affecting hardware performance. For FFT implementations under serial input/output modes, this results in significant input/output latency. This paper adopts an 8-channel parallel input/output mechanism. In conventional mode, input signals are written into storage units in the natural order of 8 parallel channels. For a 4096-point FFT algorithm, input data is sequentially arranged as shown in the left of Fig. 4, with all data represented in octal format. The first group of data (0000-0007) is written into RAM0-RAM7, the second group (0010-0017) into RAM0- RAM7, and so on, with the 64th group (1000-1007) and 128th group (2000-2007) following the same pattern. In this mode, input data such as 0000, 1000, 2000, ..., 7000 (octal) are all written into RAM0. These eight data points are required by the butterfly multiplication unit in the first cycle but cannot be read simultaneously within one cycle, necessitating registerbased caching. Idle waiting cycles caused by this method waste hardware resources. To maintain peak

performance between memory and processing units, alleviate data congestion, and improve overall efficiency, this paper introduces a commutator structure composed of multiple multiplexers. This structure reorganizes the data input pattern based on the data read sequence, with the reorganized data storage pattern illustrated on the right side of Fig. 4

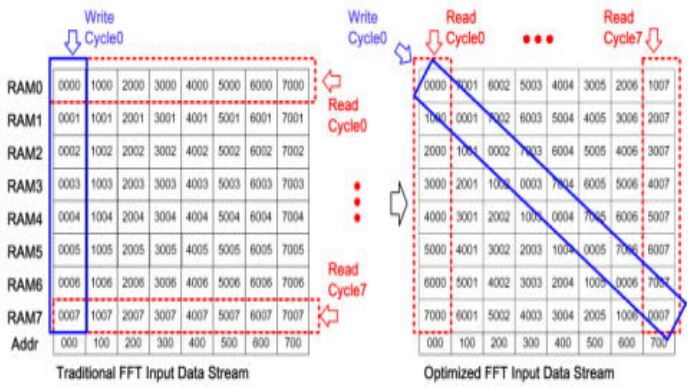


FIGURE 4. Rearrangement for 4096P input data stream.

The hardware circuit is shown in Fig. 5. The forward commutator restructures the data flow through data rotation, distributing output data across eight SRAM banks to enable single-cycle memory storage and reduce data congestion. During subsequent data read operations, the reverse commutator reverses the rotation to restore the normal data flow, ensuring uninterrupted computation in subsequent stages. In the next computation stage, the reorganized data allows direct reading of required data from different RAMs, eliminating idle cycles and effectively resolving data flow congestion for seamless read/write operations and efficient computation. Additionally, bit-reversal permutation is implicitly integrated into the commutator structure, further enhancing system performance and efficiency. After modifying the data flow, the middle data storage pattern for each computation stage is shown in Fig. 6. For processing 4096-point FFT, a 4-stage radix-8 decomposition is employed since 4096 is a power of 8. For 2048-point FFT, we utilize a combination of 3-stage radix-8 decomposition and 1-stage radix-4 decomposition.

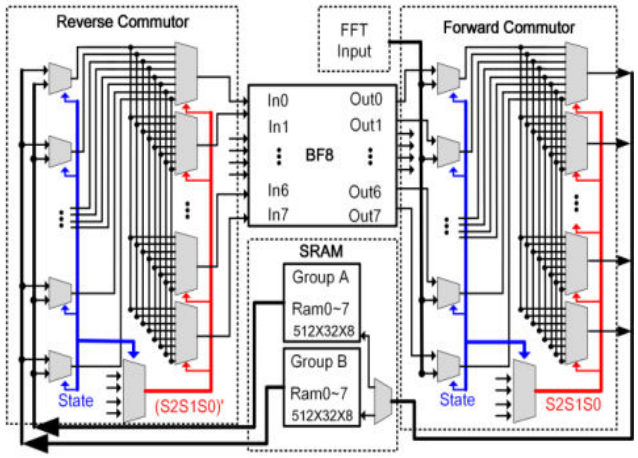


FIGURE 5. The architecture of commutator set unit.

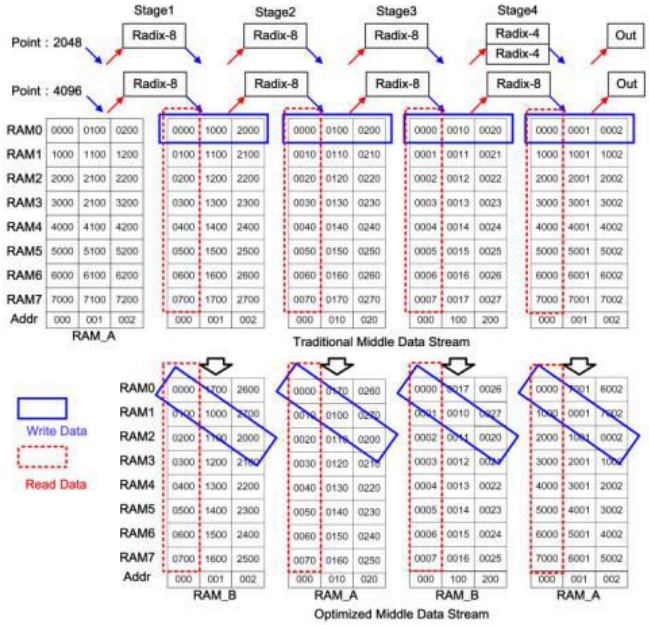


FIGURE 6. Rearrangement for FFT middle data stream.

3.3.MIXED RADIX BUTTERFLY MULTIPLICATION UNITS

Computational complexity serves as a critical metric for assessing the performance of FFT algorithms. Currently, common FFT algorithms include radix-2, radix-4, radix-8, and mixed-radix methods. The arithmetic operations required by FFT algorithms with different radix decompositions are summarized in the table below:

TABLE 1. FFT algorithm complexity comparison.

Arithmetic	Stage	Complex multiplication	Complex addition
Radix-2	$Log_2 N$	$\frac{N}{2}Log_2 N - \frac{N}{2}$	$N\,Log_2 N - N$
Radix-4	$\frac{1}{2}Log_2 N$	$\frac{3N}{8}Log_2 N - \frac{3N}{4}$	$\frac{3N}{2}Log_2 N - 3N$
Radix-8	$\frac{1}{3}Log_2 N$	$\frac{7N}{24}Log_2 N - \frac{7N}{8}$	$\frac{7N}{3}Log_2 N - 7N$
Radix-16	$\frac{1}{4}Log_2 N$	$\frac{15N}{64}Log_2 N - \frac{15N}{16}$	$\frac{15N}{4}Log_2 N - 15N$

As illustrated in Table 2, the computational complexity decreases with increasing decomposition radix. For 2048- point and 4096-point FFTs, adopting radix-8 decomposition significantly reduces the number of computational stages, thereby lowering the overall computational overhead. Although radix-16 decomposition can further reduce the number of stages, it substantially increases hardware implementation complexity and data dependencies, leading to limited gains in actual efficiency. The structure of the conventional radix-8 decomposition algorithm is depicted in Fig. 7. Building upon this, the butterfly multiplication unit is further optimized through the introduction of logical combination techniques to reduce computational complexity. Specifically, for 2048- point FFT processing, a mixed-radix computational architecture is proposed: dynamically switching between radix-4 and radix-8 computational units across different stages of the algorithm. A reconfigurable mixed-radix butterfly computational unit is designed. The constant coefficient √ 2/2 for the W1 8 , W3 8 , W5 8 , W7 8 is extracted. When performing the Radix-4 calculation, the part of Radix-8 represented by the dotted line is no longer calculated, as depicted in Fig. 8.

The circuit structure is as illustrated in Fig. 9. Initially, approximate multiplication is employed to obtain the product of the input and the twiddle factor. Subsequently, a Wallace tree structure is utilized to perform accumulation operations on the products. Finally, a multiplexer is used to choose the output signal value while simultaneously reducing the switching activity of the Radix-8 specific circuit, thereby lowering power consumption. When the radix-4 operation mode is selected, the butterfly multiplication unit, which is a dual-path parallel radix-4 unit, improves the utilization of hardware resources and reduces the operation time by half in this stage.

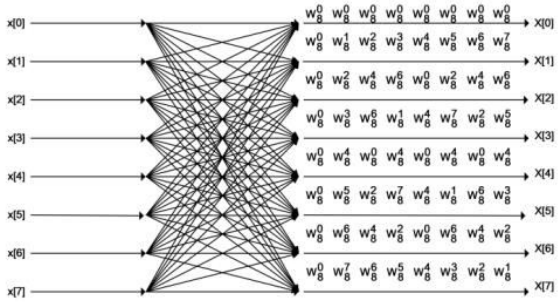


FIGURE 7. The structure of the conventional radix-8 decomposition algorithm.

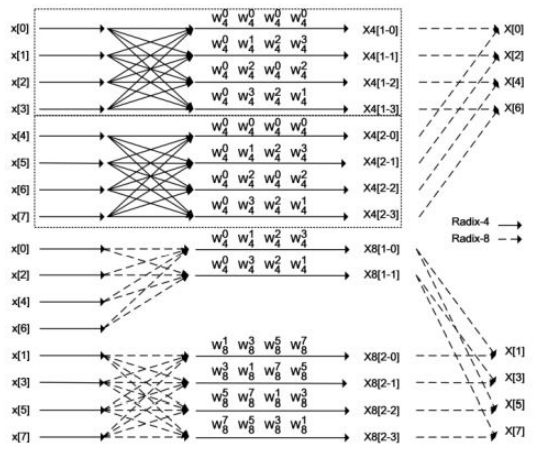


FIGURE 8. The optimized mixed radix-8 and radix-4 decomposition algorithm.

Disadvantage of Existing system

1. Limited throughput due to single-path architecture.
2. Poor support for non-power-of-two DFTs.
3. Memory access conflicts reduce performance.
4. High input/output latency.
5. Increased hardware complexity and area.
6. High power consumption during TF multiplications.
7. Limited parallel data processing capability.
8. Idle cycles cause poor hardware utilization.
9. Large memory requirement for buffering.
10. Conventional CORDIC has limited rotation range.
11. Separate scaling circuits increase cost.
12. Timing synchronization issues in pipelined stages.
13. Reduced flexibility for multimode FFT applications.

IV. PROPOSED METHOD

To design a reconfigurable CORDIC architecture with minimum reconfiguration overhead, we need to maximize the sharing of common hardware circuit in different configurations. Therefore, to explore the possibility of reconfigurable CORDIC, we examine, here, the commonalities in three main issues of CORDIC implementation, namely:

1) the coordinate-rotation matrix;

2) selection of elementary angles; and

3) direction of microrotations.

A. Reference Reconfigurable CORDIC

A basic design for reconfigurable CORDIC based on unified CORDIC algorithm was proposed in [12]. The major concern with the design of conventional reconfigurable architecture is the incompatibility in RoC of circular and hyperbolic trajectories. The RoC of circular CORDIC is [−99°, 99°], while that of hyperbolic CORDIC is given by $|\theta| \leq$ 1.1182 radians. This limits the maximum angle of rotation of the reconfigurable design to 64°. The incompatible RoC of circular and hyperbolic CORDICs makes it difficult to implement them in the same circuit to perform rotation through [−180°, 180°]. Another major issue with the conventional reconfigurable CORDIC is scaling. We need to have two different scaling circuits for circular and hyperbolic CORDIC, and select

the output from one of the scaling circuits depending on the selection of trajectory of operation.

B. Design Strategy for Proposed Reconfigurable CORDIC

As discussed in Section III-A, the circular and hyperbolic CORDICs require two different scaling circuits, which is quite costly. Therefore, it is necessary to use a scale-free implementation in the reconfigurable CORDIC. Here, we discuss the scaling-free CORDIC and its limitations, followed by the discussions on our design strategy for a reconfigurable CORDIC.

1) Scaling-Free CORDIC Algorithm and Its Limitations: The scaling-free CORDIC [2] employs second-order Taylor series approximation, where the rotation matrix is given by Ri = 1 − 2−(2i+1) −2−i 2−i 1 − 2−(2i+1) . (3) This approximation imposes a restriction on the basic-shift1 i = ———— (b − 2.585/3). For 16-bit applications, the basic-shift is i = 4, which reduces the RoC to 7.16°, which can be extended to 22.5° using multiple iterations corresponding to the basic-shift i = 4. This is a major drawback, which limits the applicability of this algorithm. Moreover, the algorithms in [2] and [3] focus only on circular rotation-mode, which cannot be directly extended to hyperbolic CORDIC, since the second order of approximation of Taylor series expansion of hyperbolic functions results in a very low RoC (nearly 22.5°). Due to the lack of symmetry in hyperbolic functions, the RoC cannot be extended to the entire coordinate space.

2) Reconfigurability of Rotation-Mode CORDIC: In [7] and [10], scaling-free algorithms for circular and hyperbolic trajectories are proposed. Moreover, in both the scaling-free algorithms, third order of approximation of Taylor series is used to derive the CORDIC rotation-matrices, as Rci = 1 − 2−(2si+1) −(2−si − 2−(3si+3)) 2−si − 2−(3si+3) 1 − 2−(2si+1) (4a) Rhi = 1 + 2−(2si+1) 2−si + 2−(3si+2) 2−si + 2−(3si+2) 1 + 2−(2si+1) . (4b) 1The minimum possible permissible shifts in the CORDIC iteration have been termed as basic-shift, which is equal to the number of right shifts in the first CORDIC iteration.

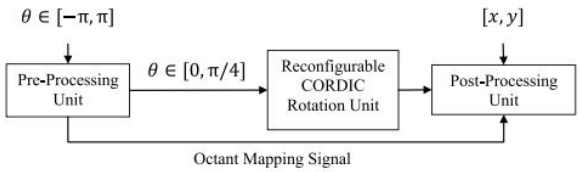


Fig. 1. Proposed reconfigurable rotation-mode CORDIC processor.

Note that the same set of elementary angles is used for both circular and hyperbolic rotation-modes. This is a big advantage to derive the reconfigurable CORDIC, since no differentiation is required to identify the microrotations according to the trajectories. For circular and hyperbolic trajectories, the elementary angles are redefined as αi = 2−si (5) where si is the number of shifts for the ith iteration. The RoC for both the trajectories is compatible and extends to the entire coordinate space. The design for rotation-mode CORDIC with slight modification can be extended to support vectoring-mode as discussed below.

3) Reconfigurability of Vectoring-Mode CORDIC: To realize a vectoring-mode CORDIC, all the microrotations will be performed in the clockwise direction for both

the circular and hyperbolic trajectories. The rotation matrices are given by Rci = 1 − 2−(2si+1) 2−si − 2−(3si+3) −(2−si − 2−(3si+3)) 1 − 2−(2si+1) (6a) Rhi = 1 + 2−(2si+1) −(2−si + 2−(3si+2)) −(2−si + 2−(3si+2)) 1 + 2−(2si+1) (6b) where si is the shift-index ith iteration. The sign-bit of the y-coordinate over successive iterations determines the angle of rotation θ. For vectoring-mode, the maximum angle of rotation that can be computed lies in the range [0, π/4]. However, this range can be extended to the entire coordinate space using octant wave symmetry of sine and cosine functions for circular trajectory

4.1.PROPOSED RECONFIGURABLE CORDIC

As seen in Sections III-B2 and III-B3, the coordinate calculation matrices for circular and hyperbolic CORDICs differ by the sign of operands, and to realize that additions are to be replaced by subtractions and vice-versa. This can be easily realized by a reconfigurable add/subtract circuit. In both cases, the basic-shift could be either 2 or 3, but the number of microrotations vary with the mode of operation. Besides, each case will have its own circuit to enable the extension of RoC. Based on these observations, we design three reconfigurable CORDIC architectures:

1) rotation-mode reconfigurable CORDIC;

2) vectoring-mode reconfigurable CORDIC; and

3) generalized reconfigurable CORDIC. A. Rotation-Mode Reconfigurable CORDIC The proposed design for reconfigurable rotation-mode CORDIC (shown in Fig. 1) consists of three parts:

1) preprocessing unit;

2) reconfigurable CORDIC rotation unit; and

3) postprocessing unit.

The preprocessing unit ensures that the input rotation angle to the CORDIC processing structure always lies in the range [0, π/4], as the maximum rotation angle that can be handled by microrotation sequence generator is π/4. The postprocessing unit is required only for circular trajectory to swap/complement the sine/cosine values depending on the octant of the rotation angle. The user can control the trajectory of the reconfigurable CORDIC by changing a 1-bit signal T . The rotation matrix for reconfigurable rotation-mode CORDIC is obtained after unifying the rotation matrices of circular and hyperbolic case given by (4a) and (4b), respectively, as Ri = 1 ± 2−(2si+1) ±(2−si ± 2−(3si+2+T)) 2−si ± 2−(3si+2+T) 1 ± 2−(2si+1) where T = 0 for hyperbolic 1 for circular. (7)

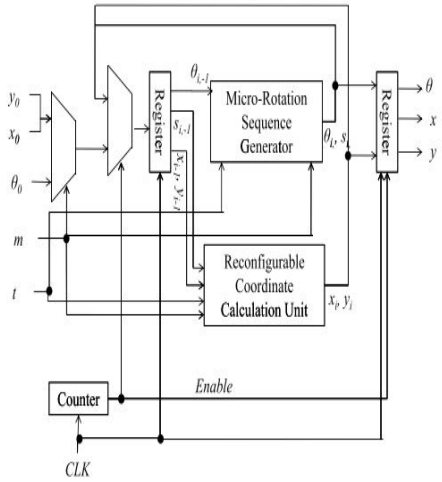


Fig. 2. Structure of the proposed reconfigurable recursive CORDIC architectures.

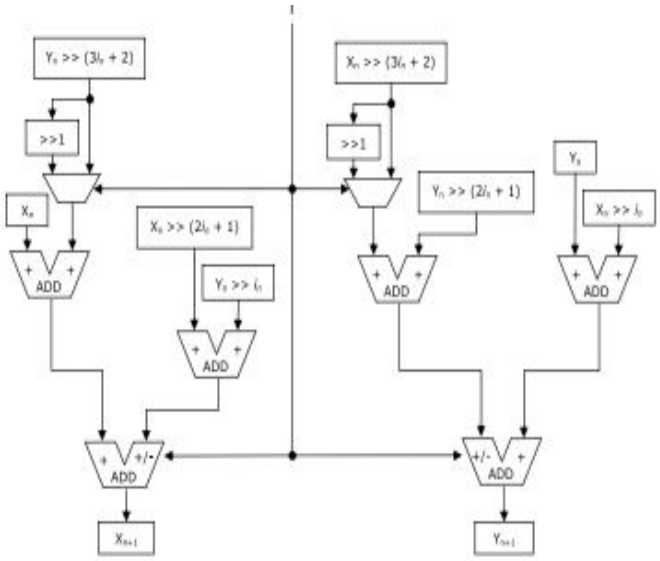


Fig. 3. RCCU for recursive design.

1) Proposed Recursive Architecture: The recursive architecture (shown in Fig. 2) uses a single CORDIC microrotator to perform all the CORDIC iterations. The circular CORDIC of [7] requires one iteration less than the hyperbolic CORDIC of [10], but here we realize the architecture for the same number of iterations (eight for sbasic = 2 and eleven for sbasic = 3) for both circular and hyperbolic trajectories. The reconfigurable coordinate calculation unit (RCCU) is shown in Fig. 3. 2) Proposed Pipelined Architecture: Fig. 4 shows the reconfigurable CORDIC rotation unit for basic-shift 2. The shift-index si is fixed in every RCCU, and hence the shifters are hardwired and do not involve high complexity barrel-shifters. The implementation of RCCUs varies according to the basic-shift si . With slight modifications, the pipeline can be extended for basic-shift 3. B. Reconfigurable Vectoring-Mode CORDIC The reconfigurable rotation matrix for vectoring mode is obtained by unifying (6a) and (6b), as Ri = 1 ± 2−(2si+1) ∓(2−si ± 2−(3si+2+T)) −(2−si ± 2−(3si+2+T)) 1 ± 2−(2si+1) where T = 0, for hyperbolic 1, for circular. (8)

By changing the implementation of the RCCU to implement (8), the recursive architecture of Fig. 2 can be used to realize CORDIC iterations for vectoring-mode. The rollover counter value is 15 for sbasic = 2, and 17 for sbasic = 3. The pipelined architecture of vectoring-mode reconfigurable CORDIC consists of eight stages for sbasic = 2, as shown in Fig. 5. Similar to reconfigurable rotation-mode CORDIC, for increasing shift-indices, the implementation of RCCUs is simplified for reconfigurable vectoring-mode CORDIC as well. The input coordinates [xin, y in] are first preprocessed to obtain coordinates [xin, yin] and octant mapping signals. The coordinates [xin, yin] are input to the vectoring-mode CORDIC pipeline to generate an angle $\theta \in [0, \pi/4]$. The rotation angle θ generated by the vectoring-mode CORDIC pipeline is mapped to the desired octant using the octant mapping signals generated by the preprocessing unit. Therefore, the RoC supported by the proposed vectoring-mode reconfigurable CORDIC is $[-\pi, \pi]$.

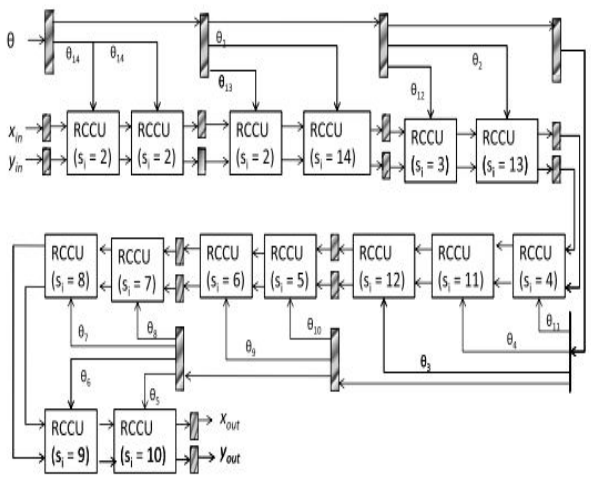


Fig. 4. Reconfigurable rotation-mode CORDIC unit for basic-shift 2.

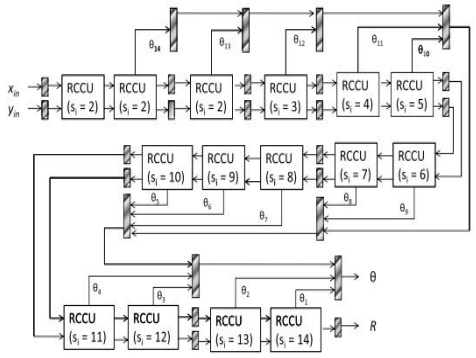


Fig. 5. Proposed pipeline reconfigurable vectoring-mode CORDIC unit for $s_{basic} = 2$.

C. Proposed Generalized Reconfigurable CORDIC The generalized reconfigurable CORDIC can operate either in vectoring-mode or in rotation-mode for both circular and hyperbolic trajectories. The user can select the trajectory of operation using a single bit signal T (T = 1 for circular and T = 0 for hyperbolic). Another single bit signal M is used to control the mode of operation (M = 0 for rotation-mode and M = 1 for vectoring-mode). The recursive architecture of the proposed generalized reconfigurable CORDIC is implemented by combining the CORDIC microrotators for both rotation-mode and vectoring-mode CORDICs, as shown in Fig. 6. The throughput of the proposed recursive generalized reconfigurable CORDIC is the same as that of the recursive reconfigurable vectoring-mode CORDIC. The block diagram for pipelined generalized reconfigurable CORDIC using basic-shift sbasic = 2 is shown in Fig. 7. It can be easily extended to basic-shift sbasic = 3 as is done for reconfigurable rotation-mode and vectoring-mode CORDICs.

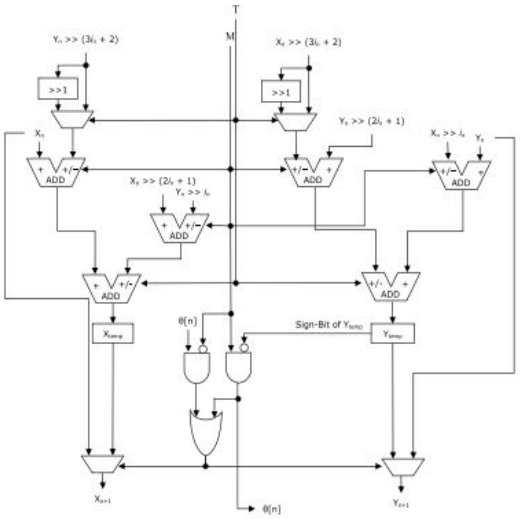
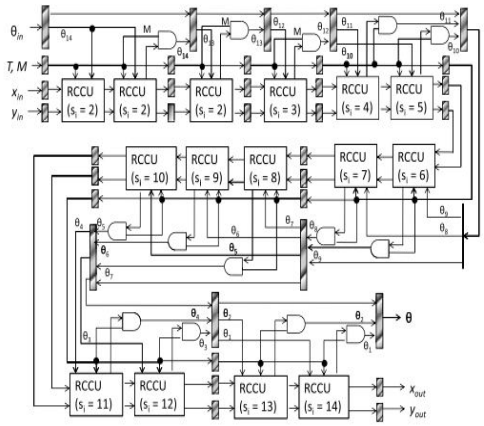


Fig. 6. Structure of CORDIC microrotator for the proposed recursive generalized reconfigurable CORDIC.



ig. 7. Proposed pipeline generalized reconfigurable CORDIC unit or $s_{basic} = 2$.

- The proposed system operates with **very low input voltage (as low as 0.35 V)**, making it suitable for energy harvesting.
- It achieves **higher output voltage (up to 2.45 V)** with improved efficiency (~58%).
- Fully integrated design eliminates the need for **external components**, reducing size and complexity.
- Requires **very small chip area (0.466 mm²)**, ideal for biomedical and implantable devices.
- Works at **low clock frequency**, resulting in reduced power consumption.
- Supports both **continuous and pulsed operation modes**, making it versatile for different applications.

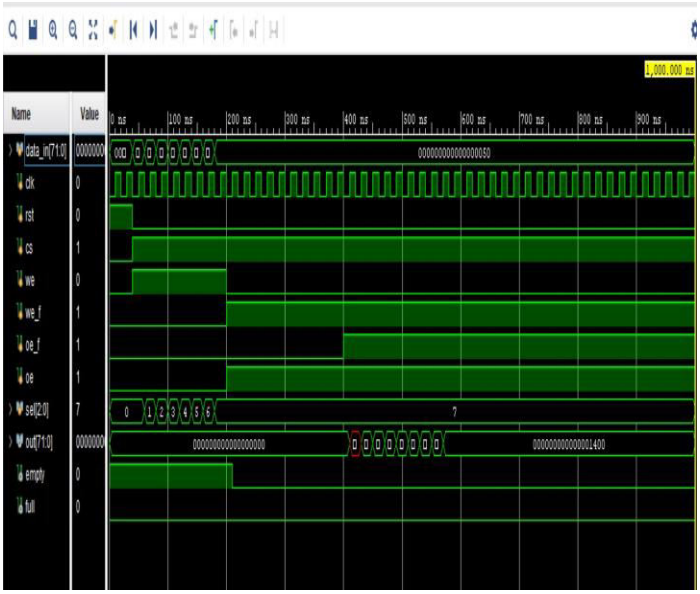


Fig :

V. RTL & SIMULATION RESULTS

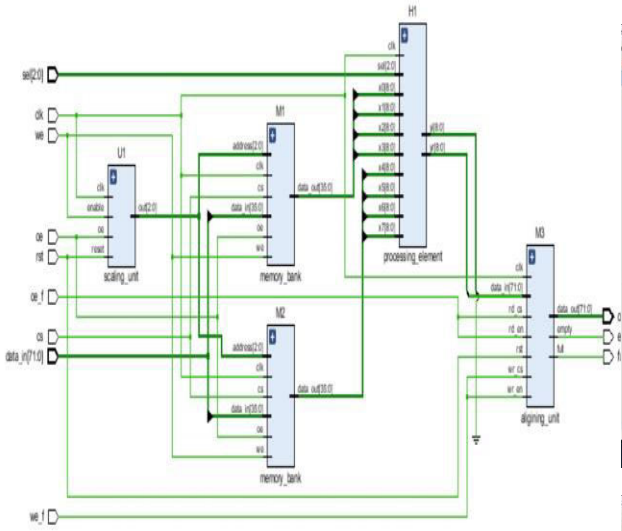
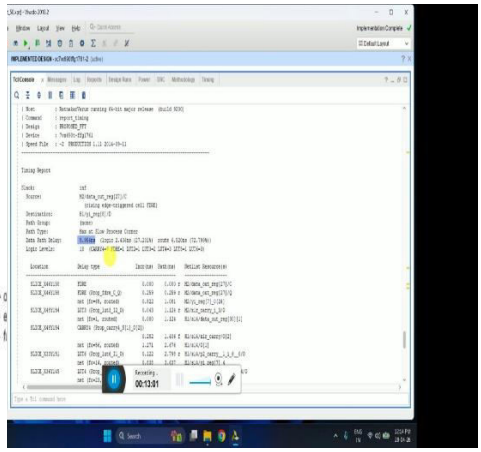
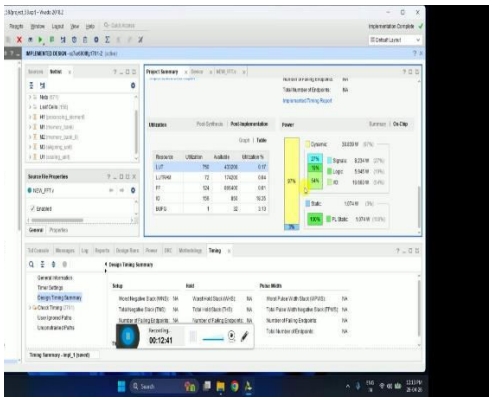
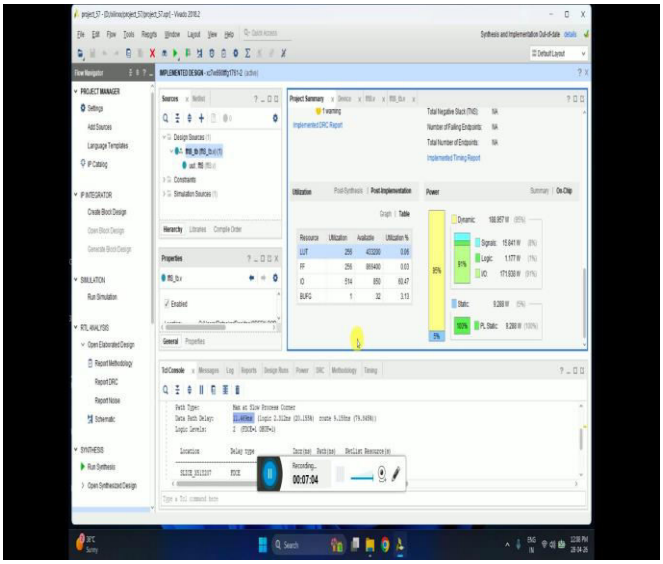


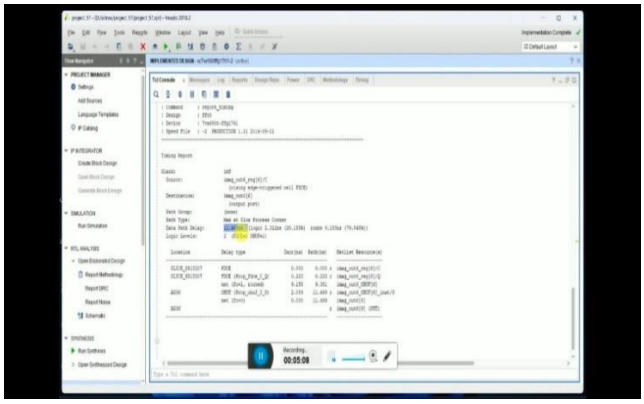
Fig :RTL





New results





VI. CONCLUSION

In this brief, for the first time a systematic design method for reconfigurable CORDIC is proposed to let a CORDIC function in different modes and different trajectories of operations. The proposed reconfigurable CORDIC architectures can be used in a variety of applications, such as synchronizers, waveform generators, low-cost scientific calculators, and so on. Approximately 60% of the area is saved by the proposed rotation or vectoring-mode reconfigurable CORDIC designs over the reference recursive reconfigurable CORDIC, without any effect on the maximum operating frequency. On the other hand, the proposed pipelined rotation and vectoring-mode reconfigurable CORDIC designs save 30%–50% area compared with the reference reconfigurable design, with nearly the same maximum operating frequency

VII. REFERENCES

[1] Y. Su, W. Shi, L. Hu, and S. Zhuang, "Implementation of SVMbased low power EEG signal classification chip," IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 69, no. 10, pp. 4048–4052, Oct. 2022, doi: 10.1109/TCSII.2022.3185309.

[2] C. Li, "A 0.61-μW fully integrated keyword-spotting ASIC with realpoint serial FFT-based MFCC and temporal depthwise separable CNN," IEEE J. Solid-State Circuits, vol. 59, no. 3, pp. 867–877, Mar. 2024, doi: 10.1109/JSSC.2023.3339528.

[3] J. Chen, K. Lin, L. Yang, and W. Ye, "An energy-efficient edge processor for radar-based continuous fall detection utilizing mixed-radix FFT and updated blockwise computation," IEEE Internet Things J., vol. 11, no. 19, pp. 32117–32128, Oct. 2024, doi: 10.1109/JIOT.2024.3422251.

[4] H. Fang, Z. Ma, F. Yu, B. Zhao, and B. Zhang, "Optimised serial commutator FFT architecture in terms of multiplexers," IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 71, no. 1, pp. 445–449, Jan. 2024, doi: 10.1109/TCSII.2023.3304696.

[5] M. Garrido and P. Paz, ‘‘Optimum MDC FFT hardware architectures in terms of delays and multiplexers,’’ IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 68, no. 3, pp. 1003–1007, Mar. 2021, doi: 10.1109/TCSII.2020.3022528.

[6] M. Garrido, ‘‘A survey on pipelined FFT hardware architectures,’’ J. Signal Process. Syst., vol. 94, no. 11, pp. 1345–1364, Nov. 2022, doi: 10.1007/s11265-021-01655-1.

[7] C. Yang, J. Wu, S. Xiang, L. Liang, and L. Geng, ‘‘A high-throughput and flexible architecture based on a reconfigurable mixed-radix FFT with twiddle factor compression and conflict-free access,’’ IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 31, no. 10, pp. 1472–1485, Oct. 2023, doi: 10.1109/TVLSI.2023.3298943.

[8] Q.-J. Xing, Z.-G. Ma, and Y.-K. Xu, ‘‘A novel conflict-free parallel memory access scheme for FFT processors,’’ IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 64, no. 11, pp. 1347–1351, Nov. 2017, doi: 10.1109/TCSII.2017.2683643.

[9] Z. Kaya and M. Garrido, ‘‘Low-latency 64-parallel 4096-point memorybased FFT for 6G,’’ IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 70, no. 10, pp. 4004–4014, Oct. 2023, doi: 10.1109/TCSI.2023.3298227.

[10] S. Liu and D. Liu, ‘‘A high-flexible low-latency memory-based FFT processor for 4G, WLAN, and future 5G,’’ IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 27, no. 3, pp. 511–523, Mar. 2019, doi: 10.1109/TVLSI.2018.2879675.

[11] Z. Kaya, M. Garrido, and J. Takala, ‘‘Memory-based FFT architecture with optimized number of multiplexers and memory usage,’’ IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 70, no. 8, pp. 3084–3088, Aug. 2023, doi: 10.1109/TCSII.2023.3245823.

[12] M. Garrido, M. Á. Sánchez, M. L. López-Vallejo, and J. Grajal, ‘‘A 4096- point radix-4 memory-based FFT using DSP slices,’’ IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 25, no. 1, pp. 375–379, Jan. 2017, doi: 10.1109/TVLSI.2016.2567784.

[13] Y. Guo, Z. Wang, Q. Hong, H. Luo, X. Qiu, and L. Liang, ‘‘A 60-mode high-throughput parallel-processing FFT processor for 5G/4G applications,’’ IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 31, no. 2, pp. 219–232, Feb. 2023, doi: 10.1109/TVLSI.2022.3227346.

[14] C. Yan, X. Zhao, T. Zhang, J. Ge, C. Wang, and W. Liu, ‘‘Design of high hardware efficiency approximate floating-point FFT processor,’’ IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 70, no. 11, pp. 4283–4294, Aug. 2023, doi: 10.1109/TCSI.2023.3298882.

[15] A. K. Y. Reddy and S. P. Kumar, ‘‘Performance analysis of 8-point FFT using approximate radix-8 booth multiplier,’’ in Proc. 3rd Int. Conf. Commun. Electron. Syst. (ICCES), Oct. 2018, pp. 42–45, doi: 10.1109/CESYS.2018.8724107.

[16] J. Du, K. Chen, P. Yin, C. Yan, and W. Liu, ‘‘Design of an approximate FFT processor based on approximate complex multipliers,’’ in Proc. IEEE Comput. Soc. Annu. Symp. VLSI (ISVLSI), Jul. 2021, pp. 308–313, doi: 10.1109/ISVLSI51109.2021.00063.

[17] P. T. L. Pereira, P. Ü. L. da Costa, G. da C. Ferreira, B. A. de Abreu, G. Paim, E. A. C. da Costa, and S. Bampi, ‘‘Energy-quality scalable design space exploration of approximate FFT hardware architectures,’’ IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 69, no. 11, pp. 4524–4534, Nov. 2022, doi: 10.1109/TCSI.2022.3191180.

[18] G. M. Santana, G. Paim, L. M. G. Rocha, R. Neuenfeld, M. B. Fonseca, E. A. C. da Costa, and S. Bampi, ‘‘Using efficient adder compressors with a split-radix butterfly hardware architecture for low-power IoT smart sensors,’’ in Proc. 24th IEEE Int. Conf. Electron., Circuits Syst. (ICECS), Dec. 2017, pp. 486–489, doi: 10.1109/ICECS.2017.8292075.

[19] G. Ferreira, P. T. L. Pereira, G. Paim, E. Costa, and S. Bampi, ‘‘A powerefficient FFT hardware architecture exploiting approximate adders,’’ in Proc. IEEE 12th Latin Amer. Symp. Circuits Syst. (LASCAS), Feb. 2021, pp. 1–4, doi: 10.1109/LASCAS51355.2021.9667154.

[20] W. Liu, Q. Liao, F. Qiao, W. Xia, C. Wang, and F. Lombardi, ‘‘Approximate designs for fast Fourier transform (FFT) with application to speech recognition,’’ IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 66, no. 12, pp. 4727–4739, Dec. 2019, doi: 10.1109/TCSI.2019.2933321.

[21] Q. Liao, W. Liu, F. Qiao, C. Wang, and F. Lombardi, ‘‘Design of approximate FFT with bit-width selection algorithms,’’ in Proc. IEEE Int. Symp. Circuits Syst. (ISCAS), May 2018, pp. 1–5, doi: 10.1109/ISCAS.2018.8350947.

[22] X.-Y. Shih, H.-R. Chou, and Y.-Q. Liu, ‘‘VLSI design and implementation of reconfigurable 46-mode combined-radix-based FFT hardware architecture for 3GPP-LTE applications,’’ IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 65, no. 1, pp. 118–129, Jan. 2018, doi: 10.1109/TCSI.2017.2725338.

[23] 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, doi: 10.1109/TCSI.2018.2860942.

[24] K.-F. Xia, B. Wu, T. Xiong, and T.-C. Ye, ‘‘A memory-based FFT processor design with generalized efficient conflict-free address schemes,’’ IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 25, no. 6, pp. 1919–1929, Jun. 2017, doi: 10.1109/TVLSI.2017.2666820.

[25] Y. Tian, Y. Hei, Z. Liu, Q. Shen, Z. Di, and T. Chen, ‘‘A modified signal flow graph and corresponding conflict-free strategy for memory-based FFT processor design,’’ IEEE Trans. Circuits Syst. II, Exp. Briefs, v