Low-Latency 64-Parallel 4096-Point Memory-Based FFT for 6G

Zeynep Kaya and Mario Garrido, *Senior Member, IEEE*

***Abstract*— This paper presents a novel 64-parallel 4096-point radix-2 memory-based fast Fourier transform (FFT) architecture for 6G. This approach is the first one to use 64 parallel branches in memory-based architectures. The challenge of designing a memory-based FFT with such a high parallelization has been accomplished by paying special attention to the large number of memories in parallel. Their control has been simplified by using the same read and write address for all of them thanks to the perfect shuffle permutation, and they are organized in groups to eliminate unnecessary registers. Likewise, a novel design for the rotation memories allows for reusing rotation coefficients among parallel rotators, and a new design for the circular counter that controls the architecture is presented. The proposed FFT architecture has been implemented on a Virtex 7 field-programmable gate array (FPGA). Experimental results reveal that the proposed architecture achieves the lowest latency in clock cycles and the highest throughput in samples per clock cycle among memory-based FFT architectures so far.**

***Index Terms*— Memory-based, parallel architecture, FFT, radix-2, low latency, 6G.**

I. Introduction

THE communication technologies are undergoing revolutionary changes with the quick growth of communication applications. Although 5G offers significant improvements with respect to previous technologies, 6G will provide a superior experience for everyone through ultra-high connectivity including both humans and machines, and hundreds of billions of devices [1], [2], [3]. This will require improvements in many applications such as autonomous vehicles [4] or telesurgery [5], which demand ultra-reliable and low-latency communications (URLLC), with latencies below 100 μs [6]. This makes latency a critical factor for 6G systems.

To achieve the low latency expected for 6G, it is crucial to reduce the latency of all the components of the communication system. Reducing the latency of any of the components will lead to a reduction of the total latency and have a direct impact on the quality of the 6G applications. For instance, a faster response of an autonomous vehicle will increase the likelihood of preventing an accident.

Among the components of a communication system, the fast Fourier transform (FFT) is a key element used to calculate the modulation/demodulation of the symbols in the physical layer [7]. Given the low-latency demand for 6G, the design of low-latency FFT architectures has become nowadays an important research topic and the success of future 6G technologies will depend on it. As a consequence, in the last years, researchers have been trying to reduce the latency of FFT architectures and new advances for memory-based [8], [9] and pipelined [10], [11], [12] FFTs have been proposed.

In memory-based FFT architectures, one of the approaches to reduce the latency is to use high-radix algorithms [8], [13], [14]. High-radix algorithms reduce the latency by decreasing the number of iterations of the FFT and the number of clock cycles needed to calculate these iterations. This results in a shorter processing time.

Another approach to reduce the latency in both memory-based and pipelined FFTs is to increase the parallelization of the architecture. In fact, this approach has been widely used for pipelined FFTs, where highly parallel [15], [16], [17], [18], [19] and fully-parallel [20], [21], [22] FFT architectures have been presented. Nevertheless, the area of these architectures is extremely large.

Regarding memory-based FFT architectures, a high parallelization is generally related to the radix, and current architectures that allow for 16-parallel data use a radix-16 butterfly [8], [13], [14], [23]. A higher radix would increase the parallelization further, but it would also increase the area of the FFT considerably. In fact, a parallelization larger than 16 has only been observed in the column FFT [24], which has a parallelization equal to the FFT size, N. However, this high parallelization is not feasible for FFTs of medium or large sizes due to the extremely large amount of resources that it needs.

Another reason why high parallelization is a challenge in memory-based architectures is the fact that a high parallelization requires a large set of memories, as these architectures use one memory per parallel branch. Thus, the control of these memories could be complex and require a large circuit area.

Manuscript received 18 April 2023; revised 20 June 2023; accepted 19 July 2023. Date of publication 1 August 2023; date of current version 29 September 2023. This work was supported in part by MCIN/AEI/10.13039/501100011033 and "ERDF A way of making Europe" under Project PID2021-126991NA-I00; in part by MCIN/AEI/10.13039/501100011033 and "ESF Investing in your future" under Grant RYC2018-025384-I; and in part by the Scientific and Technological Research Council of Turkey through the International Postdoctoral Fellowship Program under Grant 1059B192200354. This article was recommended by Associate Editor X. S. Zhang. *(Corresponding author: Zeynep Kaya.)*

Zeynep Kaya is with the Department of Electricity and Energy, Osmaneli Vocational School, Bilecik Şeyh Edebali University, 11500 Bilecik, Turkey (e-mail: zeynep.kaya@bilecik.edu.tr).

Mario Garrido is with the Department of Electronic Engineering, ETSI de Telecomunicación, Universidad Politécnica de Madrid, 28040 Madrid, Spain (e-mail: mario.garrido@upm.es).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCSI.2023.3298227.

Digital Object Identifier 10.1109/TCSI.2023.3298227

This work presents a 64-parallel 4096-point decimation-in-frequency (DIF) memory-based FFT, intended for low-latency applications in 6G. To the best of the authors' knowledge, the proposed architecture is the first one to achieve a parallelization of 64 so far. In order to solve the latency challenge for memory-based FFTs, the proposed approach uses a highly parallel radix-2 architecture that avoids high radixes and, therefore, removes the need for extremely large circuits. Furthermore, all the parallel memories are reduced to only four, which is achieved by making groups of memories with the same read and write addresses, and embedding each group in a single memory. This not only reduces the number of memories significantly but also provides a drastic simplification of the control circuit. In fact, grouping the memories also allows for removing registers that appear in the parallel paths. Given the high parallelization of the architecture, the elimination of registers in multiple parallel paths has a significant impact on the total area of the circuit. In order to control the architectures and provide conflict-free access, read and write addresses are generated by a circular counter. In this work, a novel and more advanced design than the circular counter in [25] and [26] is presented, which avoids combinational loops. As a result, the proposed architecture achieves the lowest latency in clock cycles and the highest throughput in samples per clock cycle among 4096-point memory-based FFTs so far, which makes it a valuable design for 6G communications.

We have structured the rest of the paper as follows: In Section II, we review previous concepts needed to understand the proposed approach. In Section III, we present the derivation of 64-parallel memory-based FFT architecture and describe the design of the different parts of the architecture. In section IV, we provide the experimental results and compare our design to the previous state-of-the-art FFTs. Finally, in Section V, we summarize the main conclusion of this paper.

II. Background

The N-point discrete Fourier transform (DFT) of an input complex signal $x[n]$ is calculated as

$$X[k] = \sum_{n=0}^{N-1} x[n] \cdot W_N^{nk}, \qquad k = 0, \ldots, N-1, \tag{1}$$

where $X[k]$ is the output at frequency k, and $W_N^{nk} = e^{-j\frac{2\pi}{N}nk}$. To calculate the DFT faster, different fast Fourier transform (FFT) algorithms have been proposed. One of them is the Cooley-Tukey algorithm [27]. For power-of-two FFT sizes, this algorithm calculates the FFT along $n = \log_2 N$ stages and reduces the complexity from $\mathcal{O}(N^2)$ in the DFT to $\mathcal{O}(N \log_2 N)$ in the FFT. Each FFT stage, $s = 1, \ldots, n$, consists of butterflies, which calculate sums and subtractions, and rotations in the complex plane. In memory-based FFTs, butterflies and rotators are usually grouped into processing elements (PEs) [26], which are the part of the architecture where the mathematical operations of the FFT are calculated.

In order to identify data along the FFT stages, an index

$$I \equiv b_{n-1}, \ldots, b_1, b_0, \tag{2}$$

can be defined [17], where ($\equiv$) is used throughout the paper to relate a number with its binary representation. Based on this index, it has been observed that, at each stage of the FFT, butterflies operate on pairs of data that differ in b_{n-s} [28].

A. Bit-Dimension Permutations

Bit-dimension permutations are used to reorder data along FFT stages. A bit-dimension permutation considers a set of $N = 2^n$ data, whose position in the data flow is defined from 0 to $2^n - 1$ according to [29]

$$\mathcal{P} = \sum_{i=0}^{n-1} x_i 2^i, \tag{3}$$

where $x_{n-1}, x_{n-2}, \ldots, x_0$ are called dimensions.

A data flow where P data arrive in parallel at different terminals has $p = \log_2 P$ parallel dimensions that correspond to $x_{p-1}, \ldots, x_0$. The remaining $n - p$ dimensions are serial and define data arriving in series to the terminals. These serial dimensions correspond to $x_{n-1}, \ldots, x_p$. Note that the character $\mathcal{P}$ is used for the position, whereas P corresponds to the number of parallel branches in the architecture.

For $P = 2^p$ parallel branches, the terminals are defined in the range $T = 0, \ldots, 2^p - 1$ and are related to the dimensions by

$$T = \sum_{i=0}^{p-1} x_i 2^i, \tag{4}$$

being its binary representation is

$$T \equiv T_{p-1}, \ldots, T_0 = x_{p-1}, \ldots, x_0. \tag{5}$$

Likewise, the arrival time is defined as the clock cycle where a certain datum arrives relative to the clock cycle when the first P data arrive. The arrival time is defined in the range $t = 0, \ldots, 2^{n-p} - 1$ and is calculated as

$$t = \sum_{i=p}^{n-1} x_i 2^{i-p}, \tag{6}$$

being its binary representation

$$t \equiv t_{n-p-1}, \ldots, t_0 = x_{n-1}, \ldots, x_{n-p}. \tag{7}$$

Thus, we can represent the position as

$$\mathcal{P} = t|T \equiv t_{n-p-1}, \ldots, t_0|T_{p-1}, \ldots, T_0. \tag{8}$$

Note that a vertical bar (|) is used to separate serial and parallel dimensions.

In this context, a bit-dimension permutation σ permutes data in an input position $u_{n-1}, \ldots, u_p|u_{p-1}, \ldots, u_0$ into an output position $u'_{n-1}, \ldots, u'_p|u'_{p-1}, \ldots, u'_0$, i.e.,

$$\begin{aligned} \sigma(u) &= \sigma(u_{n-1}, \ldots, u_p|u_{p-1}, \ldots, u_0) \\ &= u'_{n-1}, \ldots, u'_p|u'_{p-1}, \ldots, u'_0. \end{aligned} \tag{9}$$

The perfect shuffle permutation [24] is a type of bit-dimension permutation that shifts dimensions circularly by one bit to the left, i.e.,

$$\sigma_{PS}(u_{n-1}, u_{n-2}, \ldots, u_0) = u_{n-2}, \ldots, u_0, u_{n-1}. \tag{10}$$

If the data flow includes parallel data, (10) may be written as

$$\sigma_{PS}(u_{n-1},\ldots,u_p|u_{p-1},\ldots,u_0) = u_{n-2},\ldots,u_{p-1}|u_{p-2},\ldots,u_0,u_{n-1}. \quad (11)$$

B. Memory-Based FFT Based on the Perfect Shuffle

In memory-based FFTs, memories are not only used to store the data but also to permute the data based on the read and write addresses [26]. To achieve this, it must be ensured that, at any iteration, data are written in the addresses that are emptied in the previous iteration, i.e., $W_i = R_{i-1}$ [25], [30]. This guarantees conflict-free access without using a total memory larger than the number of data.

In our previous approach [26], we presented a conflict-free access strategy based on the perfect shuffle permutation. This strategy provides the expected b_{n-s} bit at the PE inputs at each iteration. To achieve this, the approach applies three bit-dimension permutations: a serial-serial (ss) permutation, σ_{ss}, a parallel-parallel permutation (pp) σ_{pp}, and a serial-parallel permutation, σ_{sp}. The perfect shuffle permutation, represented by σ_{PS} in (10), is calculated as the composition of these three permutations, i.e., $\sigma_{PS} = \sigma_{sp} \circ \sigma_{pp} \circ \sigma_{ss}$.

The permutation σ_{ss} is defined as

$$\sigma_{ss}(u_{n-1},\ldots,u_p|u_{p-1},\ldots,u_0) = u_{n-2},\ldots,u_p,u_{n-1}|u_{p-1},\ldots,u_0. \quad (12)$$

This permutation is calculated in the memories and works for all memories in the same way. This means that the read and write addresses are the same for all memories. These addresses are obtained from the bits of a circular counter, $c_{n-p-1},\ldots,c_0$. The write address for the first FFT iteration is $W_{A_1} = c_{n-p-1},\ldots,c_0$. The read address for the first FFT iteration is obtained by rearranging the bits of the circular counter as $R_{A_1} = c_0, c_{n-p-1},\ldots,c_{n-p-2}$. After the calculations in the PE, data are written in the same memory addresses that were emptied, being $W_{A_2} = R_{A_1}$. Likewise, subsequent read and write addresses are obtained as

$$\begin{aligned} W_{A_1} &= c_{n-p-1}, c_{n-p-2},\ldots,c_0,\\ W_{A_2} &= R_{A_1} = c_0, c_{n-p-1}, c_{n-p-2},\ldots,c_1,\\ W_{A_3} &= R_{A_2} = c_1, c_0, c_{n-p-1},\ldots,c_2,\\ &\vdots\\ W_{A_{n-p}} &= R_{A_{n-p-1}} = c_{n-p-2},\ldots,c_0,c_{n-p-1}. \end{aligned} \quad (13)$$

The permutation σ_{pp} is defined as

$$\sigma_{pp}(u_{n-1},\ldots,u_p|u_{p-1},\ldots,u_0) = u_{n-1},\ldots,u_p|u_{p-2},\ldots,u_0,u_{p-1}. \quad (14)$$

This permutation routes the data from a certain terminal at the input of the permutation to another terminal at the output. As the permutation only routes the data with wires, it does not require any additional hardware component.

The permutation σ_{sp} is defined as

$$\sigma_{sp}(u_{n-1},\ldots,u_p|u_{p-1},\ldots,u_0) = u_{n-1},\ldots,u_{p+1},u_0|u_{p-1},\ldots,u_1,u_p. \quad (15)$$

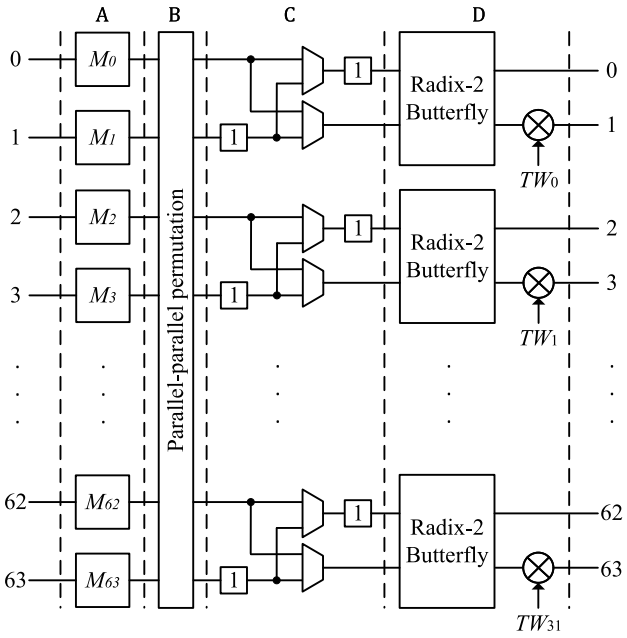

Fig. 1. 64-parallel 4096-point memory-based FFT architecture.

This permutation exchanges the lowest serial dimension and the lowest parallel dimension. It needs P registers and P multiplexers controlled with a simple signal that toggles between 0 and 1 at each clock cycle.

III. Proposed 64-Parallel 4096-Point Memory-Based FFT Architecture

Fig. 1 shows the proposed 64-parallel memory-based FFT architecture. It consists of four parts, A to D, separated by dashed lines. From A to D, these parts include a memory bank with 64 parallel memories, a parallel-parallel permutation circuit, a serial-parallel permutation circuit, and PEs that consist of radix-2 butterflies, coefficients ($TW_0,\ldots,TW_{31}$), and rotators ($\otimes$), respectively. The numbers at the left and right of the architecture represent the terminal numbers, which are defined in the range $T = 0,\ldots,63$. Each terminal T after D is connected with a wire to the terminal with the same number before A, creating the feedback connections of the memory-based architecture. These wires are not depicted, so that the figure does not become too complicated.

A. Derivation of the Memory-Based FFT Architecture for 64-Parallel Data and 4096 Points

Initially, data are stored in memories from M_0 to M_{63} in natural order, i.e., $x[0]$ to $x[63]$ are stored in the in the first addresses of memories M_0 to M_{63}, $x[64]$ to $x[127]$ in the second addresses, etc.

To formalize it mathematically, for $N = 2^n = 4096$ data, the index is defined as $I \equiv b_{11}, b_{10},\ldots,b_0$ and the initial position of the data in memory is

$$\mathcal{P}_0 \equiv \underbrace{b_{11}, b_{10},\ldots,b_6}_{\text{serial (address)}} | \underbrace{b_5,\ldots,b_0}_{\text{parallel (memory)}}, \quad (16)$$

where $b_{11}, b_{10}, \ldots, b_6$ is the address and $M \equiv b_5, b_4, \ldots, b_0$ is the memory where any datum with index $I \equiv b_{11}, b_{10}, \ldots, b_0$ is stored.

After saving data in position $\mathcal{P}_0$, the data has to be provided in the correct order to the inputs of the PEs at each stage of the FFT. Therefore, the permutation circuits and the conflict-free read and write operations must ensure that these data are in the expected order by calculating the perfect shuffle permutation. Consequently, by applying (11), we obtain the positions of the data at the PE inputs at the first stage, i.e.,

$$\mathcal{P}_1 \equiv b_{10}, \ldots, b_5 | b_4, \ldots, b_0, b_{11}. \tag{17}$$

This is fulfilled in three steps according to

$$\sigma_{PS} = \sigma_{sp} \circ \sigma_{pp} \circ \sigma_{ss}. \tag{18}$$

In the part A of Fig. 1, the serial-serial permutation (12) is applied to (16), being

$$\begin{aligned}&\sigma_{ss}(u_{11}, \ldots, u_6 | u_5, \ldots, u_0)\\ &\quad = u_{10}, \ldots, u_6, u_{11} | u_5, \ldots, u_0.\end{aligned} \tag{19}$$

This changes the order of the data read from memory with respect to the order how they were written, being the order at the input of the part B

$$\mathcal{P}_{0\mathrm{B}} \equiv b_{10}, \ldots, b_6, b_{11} | b_5, \ldots, b_0. \tag{20}$$

Note that, since the permutation permutes data only in time, there is no change in the terminals. This permutation is carried out by choosing the appropriate read and write operations of the memories according to (13). For $N = 4096$ and 64-parallel data, the circular counter has $n - p = 12 - 6 = 6$ bits. In this highly parallel architecture, the fact that the same read and write addresses are used for all the memories provides the great advantage of a simple control circuit. This also provides some simplifications for the design that are discussed later in Section III-C.

In part B of Fig. 1, the parallel-parallel permutation is carried out. Fig. 2 shows the parallel-parallel permutation circuit and the paths that data flow between terminals, $T = 0, \ldots, 63$. This permutation is calculated after the memories according to

$$\begin{aligned}&\sigma_{pp}(u_{11}, \ldots, u_6 | u_5, \ldots, u_0)\\ &\quad = u_{11}, \ldots, u_6 | u_4, \ldots, u_0, u_5.\end{aligned} \tag{21}$$

This permutation only shuffles the parallel dimensions. Therefore, σ_{pp} only routes input data to different terminals. By applying (21) to (20), the order at the input of the part C becomes

$$\mathcal{P}_{0\mathrm{C}} \equiv b_{10}, \ldots, b_6, b_{11} | b_4, \ldots, b_0, b_5. \tag{22}$$

In the part C of Fig. 1, the permutation σ_{sp} is applied and corresponds to

$$\begin{aligned}&\sigma_{sp}(u_{11}, \ldots, u_6 | u_5, \ldots, u_0)\\ &\quad = u_{11}, \ldots, u_7, u_0 | u_5, \ldots, u_1, u_6.\end{aligned} \tag{23}$$

The data are interchanged through registers and multiplexers controlled by the least significant bit (LSB) of a counter. To do the swapping by the multiplexers, data in odd branches are delayed by one clock cycle. After multiplexing operations, data at even-numbered branches are delayed one clock cycle to align the data. By applying (23) to (22), the data order at the input of part D becomes $\mathcal{P}_1$, according to (17), which is the expected order at the input of the PEs for the first FFT iteration.

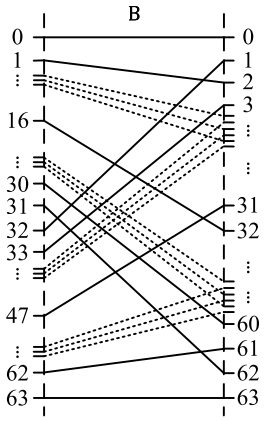

Fig. 2. Parallel-parallel permutation structure.

For the second iteration, the same perfect shuffle permutation is carried out, leading to the position

$$\mathcal{P}_2 \equiv b_9, \ldots, b_4 | b_3, \ldots, b_0, b_{11}, b_{10}, \tag{24}$$

at the input of the butterflies. Likewise, for any stage s the position at the input of the butterflies can be generalized as

$$\mathcal{P}_s \equiv \underbrace{b_{n-s-1}, \ldots, b_{n-s-6}}_{t} | \underbrace{b_{n-s-7}, \ldots, b_0, b_{n-1}, \ldots, b_{n-s}}_{T}. \tag{25}$$

It can be observed that b_{n-s} always appears in the LSB, which means that inputs to any butterfly always differ in b_{n-s}, as is required.

To clarify the data flow of the proposed approach, Fig. 3 shows an example of the data management for a 32-point 8-parallel FFT. Initially, samples are received in natural order as stated in the position $\mathcal{P}_0 \equiv b_4 b_3 | b_2 b_1 b_0$. They are written in the memories M_0 to M_7 in natural order, being $W_{A_1} = c_1 c_0$ according to (13). To start FFT operations, the first read operation is carried out according to $R_{A_1} = c_0 c_1$. Note that the reading address is the same for all the memories and is obtained according to (13). By performing read and write operations, the permutation σ_{ss} is carried out and leads the data position $\mathcal{P}_{0\mathrm{B}} \equiv b_3 b_4 | b_2 b_1 b_0$ at the input of part B. It should be noted that, so far, data are only permuted in series, i.e., data from different parallel paths are not mixed.

Position $\mathcal{P}_{0\mathrm{C}} \equiv b_3 b_4 | b_1 b_0 b_2$ shows the positions of the data at the input of part C obtained by applying the σ_{pp} permutation. As can be observed in the figure, the data differ only in terminals without any change in the time dimension. This has no hardware cost, as there is nothing more than shuffling circuits that route the data to the relevant terminals. Finally, σ_{sp}, provides the expected order $\mathcal{P}_1 \equiv b_3 b_2 | b_1 b_0 b_4$ at the input of the butterfly. The combination of all the

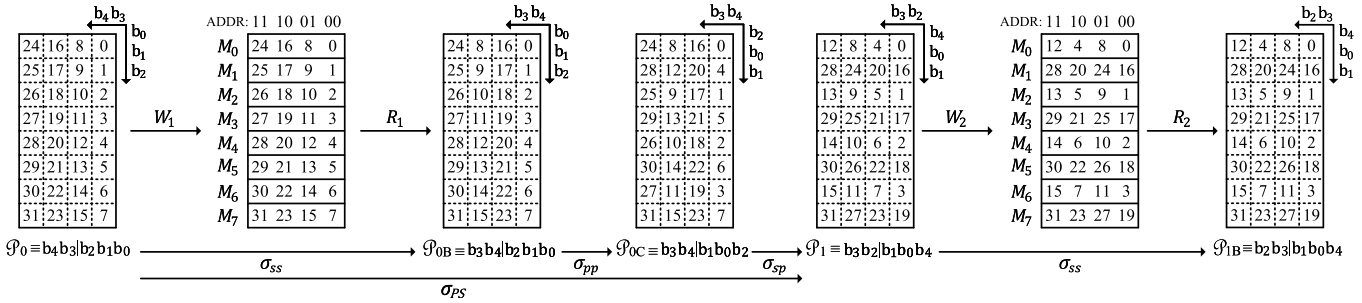

Fig. 3. Data management example for a 32-point memory-based FFT using the proposed approach.

permutations so far results in the perfect shuffle permutation of all the data in the first iteration of the FFT, i.e., σ_{PS}. After butterfly and rotation operations, σ_{PS} is carried out for the second iteration in the same way. The figure shows the second write ($W_{A_2} = c_0c_1$) and read ($R_{A_2} = c_1c_0$) operations that lead to σ_{ss}. This results in $\mathcal{P}_{1B}$, at the input of part B for the second iteration. Note that data are written in the same memory addresses that were emptied in the first iteration, i.e., as $W_{A_2} = R_{A_1} = c_0c_1$, the writing order is the same as the reading order in the first iteration. This guarantees the conflict-free access, as data are only written in addresses that are already empty.

B. Design of the Rotators

At each stage of the FFT architecture, data after the butterflies must be rotated by different angles. This requires to generate the sine and cosine coefficients of the angles. Each rotation is represented by a number, ϕ, that corresponds to a multiplication by

$$W_N^{\phi} = e^{-j\frac{2\pi}{N}\phi}. \tag{26}$$

For a radix-2 DIF FFT algorithm, the rotation values at any FFT stage, s, are calculated as a function of the index I in (2) as [31],

$$\phi_s(I) = b_{n-s} \cdot [b_{n-s-1}, \ldots, b_0] \cdot 2^{s-1}, \tag{27}$$

where $[\cdot]$ is used to indicate that the values inside are the digits of a binary number. Therefore, for the proposed architecture, the rotation values are calculated as

$$\begin{aligned}
\phi_1(I) &= b_{11} \cdot [b_{10}, \ldots, b_0] \cdot 2^0,\\
\phi_2(I) &= b_{10} \cdot [b_9, \ldots, b_0] \cdot 2^1,\\
\phi_3(I) &= b_9 \cdot [b_8, \ldots, b_0] \cdot 2^2,\\
&\vdots\\
\phi_{11}(I) &= b_1 \cdot [b_0] \cdot 2^{10},\\
\phi_{12}(I) &= 0.
\end{aligned} \tag{28}$$

To design the rotators of the architecture, we have to express the previous equations in terms of the terminal and the arrival time of the data. For instance, for the first iteration, the position of the data is provided as (17). By taking into account (17) and (8), we can observe that $t_{n-p-1}, \ldots, t_0 = t_5, \ldots, t_0 = b_{10}, \ldots, b_5$ and $T_{p-1}, \ldots, T_0 = T_5, \ldots, T_0 = b_4, \ldots, b_0, b_{11}$. Therefore, by substituting the b_i bits by the corresponding t_i and T_i bits in (28), we obtain $\phi_1(I) = T_0 \cdot [t_5t_4t_3t_2t_1t_0T_5T_4T_3T_2T_1]$ for the first iteration. The application of the same procedure to all the iterations results in

$$\begin{aligned}
\phi_1(I) &= T_0 \cdot [t_5t_4t_3t_2t_1t_0T_5T_4T_3T_2T_1],\\
\phi_2(I) &= T_0 \cdot [t_5t_4t_3t_2t_1t_0T_5T_4T_3T_20],\\
\phi_3(I) &= T_0 \cdot [t_5t_4t_3t_2t_1t_0T_5T_4T_300],\\
\phi_4(I) &= T_0 \cdot [t_5t_4t_3t_2t_1t_0T_5T_4000],\\
\phi_5(I) &= T_0 \cdot [t_5t_4t_3t_2t_1t_0T_50000],\\
\phi_6(I) &= T_0 \cdot [t_5t_4t_3t_2t_1t_000000],\\
\phi_7(I) &= T_0 \cdot [t_5t_4t_3t_2t_1000000],\\
\phi_8(I) &= T_0 \cdot [t_5t_4t_3t_20000000],\\
\phi_9(I) &= T_0 \cdot [t_5t_4t_300000000],\\
\phi_{10}(I) &= T_0 \cdot [t_5t_4000000000],\\
\phi_{11}(I) &= T_0 \cdot [t_50000000000],\\
\phi_{12}(I) &= T_0 \cdot [00000000000].
\end{aligned} \tag{29}$$

Note that $\phi_{12}(I) = 0$, so no rotation has to be calculated at the last stage.

In (29), the t_i bits only have information about the time of arrival of the data, whereas T_i only have information about the terminal. This allows us to draw several conclusions. First, all the ϕ values multiply a number by T_0. Thus, if $T_0 = 0$, no rotation must be carried out. As T_0 is the LSB of the terminal, it distinguishes even and odd terminals, being $T_0 = 0$ for even ones. Therefore, in even terminals, no rotation must be calculated. This is why in part D of Fig. 1 rotators only appear in odd terminals. This is relevant for the area of the architecture, as rotators are only needed in half of the 64-parallel branches.

Second, for stages 1 to 5 the value ϕ depends on the terminal, so it is different for different rotators. However, for stages 6 to 11, all the rotators receive the same coefficients. In fact, in stage 1 all the 32 rotators have different rotation values. In stage 2, there will be 16 different configurations, as $\phi_2(I)$ depends on 4 T_i bits, and not on T_1. Likewise, for any stage $s = 1, \ldots, 5$, there are 2^{6-s} different configurations. Additionally, the configuration of the upper rotator, for which $T \equiv T_5, \ldots, T_0 = 000001$, can be used for several rotators at stages 2 to 5. For instance, at stage 4 it can be used for any rotator for which $T_5, \ldots, T_0 = 00XXX1$, where the Xs can take any value 0 or 1. By generalizing this idea, it can be observed that the 32 sets of coefficients needed in the first stage can provide the rotations for any rotator at any other

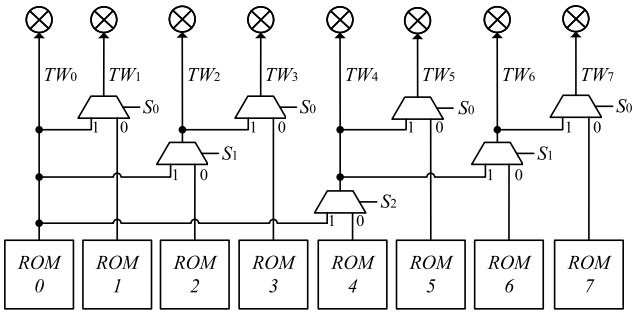

Fig. 4. Generation of rotation coefficients.

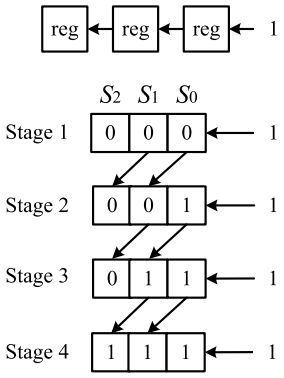

Fig. 5. Generation of the selection signals in Fig. 4.

stage. Each of these sets of coefficients is stored in a read-only memory (ROM) and Fig. 4 shows the circuit that routes the data from the memories to the rotators. This routing is done by the selection signals S_2, S_1 and S_0. For simplicity, the circuit considers the case of 8 rotators, but it is generalized easily to the case of 32 rotators for the proposed architecture, where five signals $S_4, \ldots, S_0$ are used.

Fig. 5 shows the generation of the selection signals used in Fig. 4. For stage 1, all the signals are equal to 0. Thus, each rotator takes the rotation coefficients from a different ROM memory. For stage 2, $S_0 = 1$ and the remaining selection signals are equal to zero. This makes pairs of consecutive rotators take the same rotation coefficients. In general, for a stage s, the selection signals $S_{s-2}, \ldots, S_0$ are set to 1 whereas the other selection signals are set to 0. This way, each rotator receives the coefficient that it requires at each iteration.

The selection signals are generated by a simple shift register (reg) that is initialized with zeros for the first stage. At each successive stage, a left shift operation is carried out and a 1 is added to the LSB.

Third, each rotation memory has to store 64 rotation coefficients. This is due to the fact that the ϕ values depend on the 6 bits of the arrival time, being $64 = 2^6$. For stages 1 to 6, a different value is read from the memories at each clock cycle, as all the t_i bits are used to generate the ϕ values. For stages 7 to 11, the ϕ values only depend on some of the bits of t_i. Therefore, only some of the addresses of the memory must be read.

Considering the rotation values in (29), the values T_i distinguish the memories and the values t_i distinguish the time when the memories are read. Thus, the latter lead to different

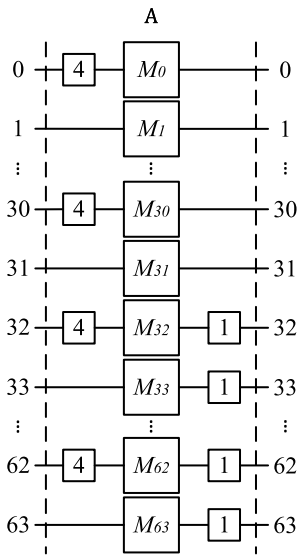

Fig. 6. Data path with memories and delay registers.

addresses of the memories. As a consequence, the coefficients stored in address a of memory m are

$$\left[\cos\left(\frac{2\pi}{N}(32a+m)\right), -\sin\left(\frac{2\pi}{N}(32a+m)\right)\right], \quad (30)$$

where a ranges from 0 to 63 and m ranges from 0 to 31, which refers to the 32 coefficient memories of the architecture.

It is worth noting that the read address for all the rotation memories is the same, so they can be embedded in a single memory, which simplifies the hardware.

As a result, this strategy of reusing the rotation memory among different stages and rotators, not only simplifies the control but also allows to use only a total rotation memory size of $64 \cdot 32 = 2048$, which corresponds to $N/2$.

Finally, for the implementation of the rotators we have used the low-area complex multiplier proposed in [32]. This implementation uses only 3 DSP slices and no slice, and calculates the rotation in only 4 clock cycles.

C. Reduction of the Registers by Adapting Read and Write Addresses

In our design, 64 memories are used in parallel, and the read and write addresses are generated according to (13). Although all the read and write addresses are the same, some delay registers on the paths cause differences in read and write times for data that arrive at different memories.

Fig. 6 shows all delay registers before and after memories. These delays are caused by the rotators and the serial-parallel circuits. On the one hand, rotators are included in odd-numbered paths, which are the paths that follow the lower outputs of the butterflies. This takes 4 clock cycles and causes delays on paths $1, 3, 5, \ldots, 63$. To equalize delays among paths and be able to perform write and read operations simultaneously in all memories, it is required to add 4 registers in the even-numbered paths, $0, 2, 4, \ldots, 62$, which do not have rotators. These registers used to equalize the timing are shown to the left of the memories in Fig. 6. This demands to use of

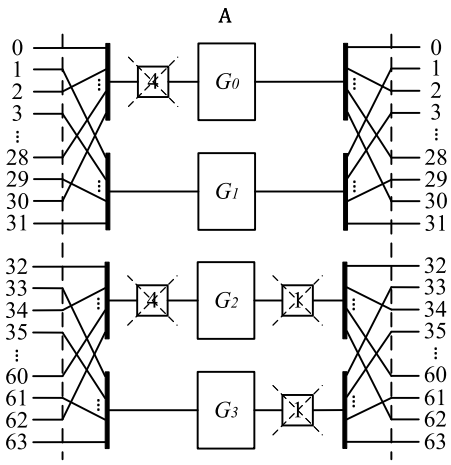

Fig. 7. Data management of memory groups.

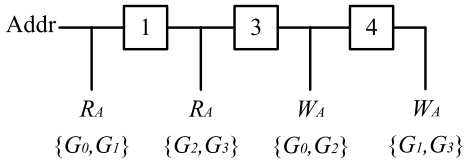

Fig. 8. Timing of read and write addresses to the different memory groups.

$4 \times 32 = 128$ additional registers of $16 + 16 = 32$ bits each. On the other hand, there are 1-delay registers on the paths 32 to 63 that come from the serial-parallel circuits. Note that these registers appear in odd branches in part C of the architecture. However, after moving them back through the parallel-parallel permutation circuit, they end up in branches 32 to 63 close to the memories. This demands to use $1 \times 32 = 32$ registers of $16 + 16 = 32$ bits each. As a result, the total number of 1-bit registers equals $(128 + 32) \times 32 = 160 \times 32 = 5120$. This increases the hardware cost significantly. Note that in an FFT with small parallelization, the impact of these registers would not be high. However, as the parallelization of our architecture is high, the total amount of registers becomes significant.

To mitigate the excess of hardware due to the registers before and after memories, we have classified the memories into groups. Fig. 7 shows the groups of memories, G_0, G_1, G_2, G_3, and the data paths with delay registers. Thus, the memories are simply placed in 4 groups where each group includes memories with the same characteristics. The groups of memories G_0 and G_2 have 4-delay registers at the input. Likewise, G_2 and G_3 have 1-delay registers at the outputs. We propose to remove these delay registers by changing the timing of the read and write addresses.

Fig. 8 shows the read and write times of each memory group. The "Addr" signal is generated by a circular counter and provides relevant addresses for each clock cycle. Note that read and write addresses (R_A, W_A) are the same for all groups, but they have different timing. The exact time for each read and write address is achieved by $1 + 3 + 4 = 8$ registers. Considering that the memory addresses have 6 bits, the circuit in Fig. 8 only requires $6 \times 8 = 48$ 1-bit registers. Thus, with this approach, we reduce the number of 1-bit registers from 5120 to 48 and reduce the area considerably.

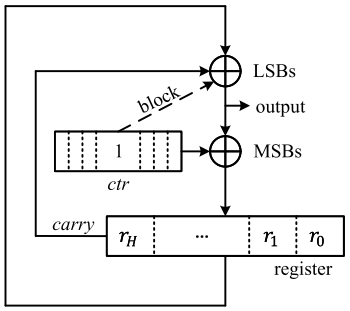

Fig. 9. Proposed circular counter.

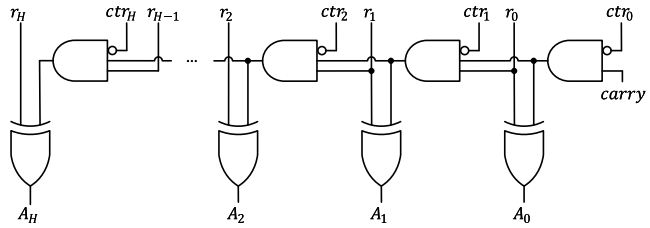

Fig. 10. LSB adder of the circular counter.

Besides the advantage of removing registers and, thus, saving hardware, this approach has another advantage. By grouping memories in 4 groups and using each group as a single memory, the design only requires 4 memories in parallel instead of 64. Furthermore, the total data memory size is equal to the FFT size, N, which in our architecture is 4096.

D. Circular Counter

In this work, we propose an evolution of the circular counter previously used in [25] and [26]. The previous version of the circular counter had the advantage of a simple design that removed the need for large multiplexers to select the bits of the counter. However, this counter has a feedback connection without registers that creates a combinational loop. This is not a problem for the design, because the logic itself impedes that the signals propagate through one entire loop. However, in digital circuits, it is preferable to avoid combinational loops, and the new circular counter is designed with this goal.

A way to remove a combinational loop is to include a register in the loop. However, this can not be done in a straightforward way in the circular counter, because the bit that acts as the LSB changes for different stages of the FFT. Thus, placing the register between two bits of the circular counter would break the result at this point. To solve this issue, the solution that we have adopted consists in calculating part of the new value of the count in a previous clock cycle and another part in the current clock cycle.

Fig. 9 shows the structure of the proposed circular counter. At each clock cycle, the value of the count is provided at the output signal. The counter is controlled by the register "ctr", which has the same size as the counter. All the bits of "ctr" are zero except one of them, which indicates the bit that acts as LSB.

The calculation of the next output is carried out in two steps. First, "ctr" is added to the "output" signal that has the current

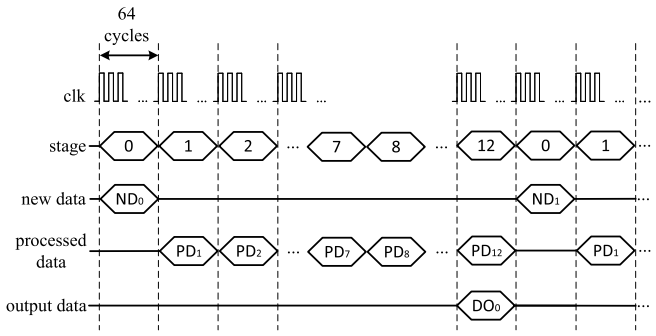

Fig. 11. Timing diagram of the proposed memory-based FFT architecture.

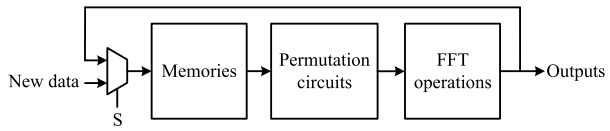

Fig. 12. Data load operation.

value of the count. This creates a sum in the most significant bits (MSBs) of the counter. As all the bits of "ctr" are zero except one, all the bits to the right of the bit that is equal to '1' will not be affected. The addition of "ctr" to the MSBs is registered in the "register" to break the combinational loop. As the counter is circular, the carry of the MSB sum must be added to the LSBs. This is done in the LSB adder. The LSB adder is a regular adder with the exception that it only adds values until it reaches the "ctr" bit set to '1'. This can be observed in Fig. 10, which shows the LSB adder. This way, the addition of the LSB adder does not propagate to the MSBs. As a result, the combination of the MSB and LSB adders covers all the bits of the counter and, therefore, implements a circular counter where the LSB bit is indicated by the "ctr" register.

E. Timing

Fig. 11 presents the timing diagram of the proposed architecture related to stages. FFT calculations are completed in $\log_2 N$ stages, which is indicated in the diagram as 1 to 12. Each stage takes $2^{n-p} = 64$ clock cycles. At stage 0, new data (ND_0) are loaded into memories before the start of the FFT operations. From stage 1 to 12, data are processed iteratively. The processing times are indicated as $PD_1, \ldots, PD_{12}$. It can be observed that the FFT results (DO_0) can be provided during the last processing stage. As there is no need to save the data in the memory at the last stage, FFT results are obtained at the same time, without the need for extra time. After the results of an FFT are provided at the output, the same procedure repeats for the next FFT.

Fig. 12 shows the data load operation. This operation is controlled with a control signal, S. This signal is activated at the beginning of the stage 0 so that new data are written into the memories. Note that, in this case, data are written in all memories simultaneously, being the writing address the same for all of them. Therefore, only one address is used for writing data during the loading phase.

The latency and throughput are key performance indicators for FFTs. The latency is defined as the number of clock cycles that the architecture needs to calculate the FFT of an input sequence, whereas the throughput indicates the average number of samples per clock cycle that are processed. If we denote the loading time as T_{LOAD}, the processing time as T_{PROC}, and the time to output results as T_{OUT}, then, the latency is calculated as

$$T_{LAT} = T_{LOAD} + T_{PROC} + T_{OUT}. \tag{31}$$

In the proposed approach, $T_{OUT} = 0$, because the output results are provided during the last processing stage. Likewise, $T_{LOAD} = N/P$ and $T_{PROC} = N/P(\log_2 N)$. Therefore, the latency of the proposed architecture in clock cycles is

$$T_{LAT} = \frac{N}{P} + \frac{N}{P}\log_2 N, \tag{32}$$

and the latency in μs can be obtained as

$$T_{LAT}(\mu s) = \frac{T_{LAT}}{f_{CLK}(\mathrm{MS}/s)}, \tag{33}$$

where f_{CLK} is the clock frequency.

The throughput in samples per clock cycle is calculated as the number of samples processed in an FFT, N, divided by the time difference between two consecutive FFTs, i.e.,

$$\mathrm{Th} = \frac{N}{\Delta T_{FFT}}. \tag{34}$$

In the proposed architecture, $\Delta T_{FFT} = T_{LOAD} + T_{PROC}$, which is the time between the beginning of ND_0 and ND_1. By substituting it in (34), we obtain

$$\mathrm{Th} = \frac{P}{1 + \log_2 N}. \tag{35}$$

The throughput in megasamples per second (MS/s) is then calculated as

$$\mathrm{Th}(\mathrm{MS}/s) = \mathrm{Th} \times f_{\mathrm{CLK}}(\mathrm{MS}/s). \tag{36}$$

F. Hardware Components

The proposed 64-parallel 4096-point radix-2 memory-based FFT architecture uses a total memory size of $4096 + 32$ for data. This includes four memory groups that implement 16 memories of 64 addresses each, and 32 registers that correspond to those after the multiplexers in the serial-parallel permutation circuit of part C in Fig. 1. Note that the registers previous to the multiplexers in the serial-parallel circuit have been removed by changing the timing of the addresses of the memory groups, as explained in Section III-C.

The rotation memory consists of 32 memories with 64 addresses each, where all of them are embedded in a single memory thanks to the fact that the reading address is the same for all of them. This results in a total of memory size of 2048 addresses for rotation coefficients.

The proposed architecture also uses 64 multiplexers to permute data in the serial-parallel permutation circuit. Furthermore, 64 complex adders and 32 complex multipliers are used to carry out the calculations in the PEs.

TABLE I

COMPARISON OF 4096-POINT MEMORY-BASED FFT ARCHITECTURES

Parameter	[33]	[8]	[26]	[23]	[13]	Proposed
FFT Type	Memory-based	Memory-based	Memory-based	Memory-based	Memory-based	Memory-based
N	4096	4096	4096	4096	4096	4096
P	4	16	4	16	16	64
Radix	2	16	2	8	4^2	2
Iterations	11	3	12	4	3	12
Word length (WL) (bits)	-	14	16	21	16	16
Technology	Virtex 7	SMIC (65nm)	Virtex 7	TSMC (12nm)	TSMC (90nm)	Virtex 7
Data type	Real	Complex	Complex	Complex	Complex	Complex
IOC (cycles)	1024	266	1024	256	256	64
CC (cycles)	11265	768	12288	1024	-	768
Latency (cycles)	12289	1054	13312	1320	-	832
Throughput (samples/cycle)	0.36	3.89	0.30	3.10	2.66	4.92
SQNR (dB)	-	66.1	45.3	82.6	70.0	44.8
Adders	2	358*	4	52	32	64
Multipliers	1	45*	2	13	8	32
Memory	4096	4096	$4096+4$	4096	$4096+32$	$4096+32$
Multiplexers	16	128	4	49	24	64

-: Not available. IOC: Number of cycles to input and output data.

*: Real adder and real multiplier. CC: Number of computation cycles in an FFT.

TABLE II

COMPARISON OF 4096-POINT MEMORY-BASED FFTS ON VIRTEX 7 FPGA

Parameter	[33]	[26]	Proposed
FFT Type	Memory-based	Memory-based	Memory-based
N	4096	4096	4096
P	4	4	64
Radix	2	2	2
Iterations	11	12	12
Word length	-	16	16
Device	Virtex 7	Virtex 7	Virtex 7
f_{CLK} (MHz)	410	342	300
Latency (μs)	30	39	2.7
Th.(MS/s)	136	105	1476
Slices	-	210	2319
LUTs	2863	465	6800
FFs	2992	641	7027
DSP slices	24	8	96
BRAMs	8	6	32
Power (mW)	-	208	1754
Power efficiency (MS/s/mW)	-	0.50	0.84

-: Not available.

G. Generalization for Other FFT Sizes and Parallelization

By following the same derivation as for the 4096-point 64-parallel FFT, the proposed approach can be generalized to any number of parallel branches, $P = 2^p$, and any power-of-two FFT size, $N = 2^n$.

A P-parallel architecture consists of P memory banks for data in parallel. Each of them has N/P addresses. This results in a total memory of N for data. As rotators only appear in odd branches, the coefficient memory includes $P/2$ banks. Each of these banks has N/P addresses, leading to a total coefficient memory of size $N/2$. By taking into account the $P/2$ registers that come from serial-parallel permutation circuits, the total size for registers and memory in the architecture is $3N/2 + P/2$.

Besides, the architecture has P multiplexers and $P/2$ PEs. Each PE consists of a butterfly that includes one complex adder and one complex subtractor, which have the same complexity and are counted as adders, followed by a complex rotator. This results in P complex adders and $P/2$ complex rotators for the entire architecture.

Finally, the total processing time is $N/P(\log_2 N)$, and the latency and throughput are calculated according to (31) and (34), respectively.

IV. EXPERIMENTAL RESULTS AND COMPARISON

The proposed architecture has been implemented on a Virtex 7 XC7VX330T -1 FFG1157 FPGA using Vivado 2021.2. It works at 300 MHz, has a latency of 832 cycles and its throughput is 4.92 samples per cycle. Additionally, the signal-to-quantization-noise ratio (SQNR) is 44.8 dB.

In Table I, we compare the proposed architecture to previous 4096-point memory-based FFT architectures. Some approaches in the table are implemented on a Virtex 7 FPGA [26], [33], including the proposed approach, whereas other approaches are implemented on ASICs [8], [13], [23]. To make the works comparable, the table includes figures of merit that are independent of the technology. Compared to the 16-parallel high-radix architectures in [8] and [23], the proposed approach achieves a latency 26% and 21% lower, respectively. In addition, we obtain 26%, 58% and 84% higher throughput compared to [8], [13], and [23], respectively. Moreover, if we examine the 4-parallel radix-2 approaches [26], [33], it can be observed that the proposed architecture has at least 14.7 times lower latency and 13.6 times higher throughput, at the cost of a larger number of hardware components.

TABLE III

AREA AND POWER BREAKDOWN OF THE SUBCOMPONENTS IN THE PROPOSED ARCHITECTURE

Parameter	Butterfly	Memory	Rot.	Perm.	Control
Slices	923	748	1057	619	150
LUTs	2561	2048	1979	0	213
FFs	2048	31	1727	1024	149
DSP slices	0	0	96	0	0
BRAMs	0	32	0	0	0
Power (mW)	236	732	587	45	50

Regarding signal-to-quantization-noise ratio (SQNR), the proposed approach achieves an SQNR of 44.8 dB. This is a typical value for a word length of 16 bits. Other approaches in the comparison report higher SQNR. In the case of [23] higher SQNR is achieved by using a larger word length. In the case of [8] and [13], the higher SQNR is the result of using block floating-point (BFP) representation for the numbers.

Table II provides the experimental results of the proposed approach in terms of clock frequency (f_{CLK}), latency, throughput (Th.), area, and power consumption. These results are compared to previous 4096-point FFTs implemented on Virtex 7 FPGAs. The proposed architecture works at a clock frequency of 300 MHz with a latency of 2.7 μs and a throughput of 1476 MS/s. The architecture uses a total of 96 DSP slices for rotations, 32 BRAMs to store the data and coefficients, 2319 slices, and its power consumption is 1754 mW.

To compare the area-time trade-off we have considered the reduction in latency of the proposed architecture and the increase in area with respect to previous approaches. Compared to [33], the proposed architecture reduces the latency by a factor of 11.1, whereas the area is increased by a factor of 2.3 in LUTs and FFs, and 4 in DSP Slices and BRAMs. Therefore, the reduction in latency is much more significant than the increase in area. Likewise, compared to [26], the proposed architecture reduces the latency by a factor of 14.4, whereas the area is increased by a factor of 11 in Slices, 12 in DSP Slices, and 5.3 in BRAMs. Thus, the latency reduction is again more significant than the increase in area.

Regarding power, the power consumption of the proposed architecture is higher than that in [26]. This comes from the extremely high throughput that the architecture achieves. However, it can be observed that the efficiency of the proposed architecture is 0.84 MS/s/mW, whereas the efficiency in [26] is 0.50 MS/s/mW, which is obtained as the throughput in samples per clock cycle divided by the power. Therefore, despite its higher power consumption, the proposed architecture is 68% more efficient than that in [26].

Even with lower clock frequency and more area than the other memory-based architectures [26], [33], the proposed one gets significantly better results in terms of latency and throughput, which sets it closer to the requirements of 6G technologies.

Finally, Table III shows the detailed area and power consumption report of the proposed architecture extracted from Vivado. The report presents the number of slices, LUTs, FFs, DSP slices, and BRAMs of each subcomponent, i.e., butterflies, memories, rotations (Rot.), which include rotators and rotation memories, permutation circuits (Perm.) and control circuits. The table shows that most of the slices are used by rotations and butterflies. Likewise, LUTs and FFs are occupied most by butterflies. Besides, DSP slices are used in only rotations, and BRAMs are only used for memories. As a consequence, the highest power consumption is due to memories and rotations, which reach 41% and 33% of the total, respectively.

V. CONCLUSION

In this paper, we have proposed a 64-parallel 4096-point radix-2 memory-based FFT architecture. This approach decreases the latency and increases the throughput by increasing the parallelization to 64, being the first time that this high parallelization is suggested for memory-based architectures.

The proposed architecture includes an efficient management of the rotations, eliminates registers before and after data memories, presents a novel circular counter, and provides a detailed analysis of the timing in the architecture.

Experimental results show that the proposed design achieves the lowest latency in terms of clock cycles and the highest throughput in samples per clock cycle among memory-based architectures so far.

ACKNOWLEDGMENT

The authors would like to thank P. Paz for providing the implementation of the rotators used in the proposed architecture.

REFERENCES

[1] C.-X. Wang et al., "On the road to 6G: Visions, requirements, key technologies and testbeds," *IEEE Commun. Surveys Tuts.*, vol. 25, no. 2, pp. 905–974, 2nd Quart. 2023.

[2] D. C. Nguyen et al., "6G Internet of Things: A comprehensive survey," *IEEE Internet Things J.*, vol. 9, no. 1, pp. 359–383, Jan. 2022.

[3] C. D. Alwis et al., "Survey on 6G frontiers: Trends, applications, requirements, technologies and future research," *IEEE Open J. Commun. Soc.*, vol. 2, pp. 836–886, 2021.

[4] S. Hakak et al., "Autonomous vehicles in 5G and beyond: A survey," *Veh. Commun.*, vol. 39, Nov. 2022, Art. no. 100551.

[5] S. Nayak and R. Patgiri, "6G communication technology: A vision on intelligent healthcare," in *Health Informatics: A Computational Perspective in Healthcare*. Singapore: Springer, Jan. 2021, pp. 1–18.

[6] N. Rajatheva et al., "White paper on broadband connectivity in 6G," 6G Res. Vis., Univ. Oulu, Oulu, Finland, Tech. Rep. 10, Jun. 2020.

[7] 3GPP. (Sep. 2020). *3GPP TS 38.211—Physical Channels and Modulation V16.3.0*. [Online]. Available: https://www.3gpp.org/DynaReport/38-series.htm

[8] 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.

[9] J. Hazarika, M. T. Khan, and S. R. Ahamed, "Low-complexity continuous-flow memory-based FFT architectures for real-valued signals," in *Proc. Int. Conf. VLSI Design Int. Conf. Embedded Syst.*, Jan. 2019, pp. 46–51.

[10] H.-J. Lin and C.-A. Shen, "The architectural optimizations of a low-complexity and low-latency FFT processor for MIMO-OFDM communication systems," *J. Signal Process. Syst.*, vol. 93, no. 1, pp. 67–78, Jan. 2021.

[11] M. Mahdavi, O. Edfors, V. Öwall, and L. Liu, "A low latency FFT/IFFT architecture for massive MIMO systems utilizing OFDM guard bands," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 66, no. 7, pp. 2763–2774, Jul. 2019.

[12] G. Prasanna Kumar, B. T. Krishna, and K. Pushpa, "Optimized pipelined fast Fourier transform using split and merge parallel processing units for OFDM," *Wireless Pers. Commun.*, vol. 117, no. 4, pp. 3067–3089, Apr. 2021.

[13] S. J. Huang and S. G. Chen, "A high-parallelism memory-based FFT processor with high SQNR and novel addressing scheme," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2016, pp. 2671–2674.

[14] S.-J. Huang and S.-G. Chen, "A high-throughput radix-16 FFT processor with parallel and normal input/output ordering for IEEE 802.15.3c systems," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 59, no. 8, pp. 1752–1765, Aug. 2012.

[15] J. Wang, C. Xiong, K. Zhang, and J. Wei, "A mixed-decimation MDF architecture for radix-2^k parallel FFT," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 24, no. 1, pp. 67–78, Jan. 2016.

[16] S.-N. Tang, J.-W. Tsai, and T.-Y. Chang, "A 2.4-GS/s FFT processor for OFDM-based WPAN applications," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 57, no. 6, pp. 451–455, Jun. 2010.

[17] M. Garrido, J. Grajal, M. A. Sánchez, and O. Gustafsson, "Pipelined radix-2^k feedforward FFT architectures," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 21, no. 1, pp. 23–32, Jan. 2013.

[18] M. Garrido, M. Acevedo, A. Ehliar, and O. Gustafsson, "Challenging the limits of FFT performance on FPGAs," in *Proc. Int. Symp. Integr. Circuits (ISIC)*, Dec. 2014, pp. 172–175.

[19] J. K. Jang, H. K. Kim, M. H. Sunwoo, and O. Gustafsson, "Area-efficient scheduling scheme based FFT processor for various OFDM systems," in *Proc. IEEE Asia Pacific Conf. Circuits Syst. (APCCAS)*, Oct. 2018, pp. 338–341.

[20] J. Hazarika, S. R. Ahamed, and H. B. Nemade, "Low-complexity, energy-efficient fully parallel split-radix FFT architecture," *Electron. Lett.*, vol. 58, no. 18, pp. 678–680, Aug. 2022.

[21] 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.

[22] G. Polat, S. Ozturk, and M. Yakut, "Design and implementation of 256-point radix-4 100 Gbit/s FFT algorithm into FPGA for high-speed applications," *ETRI J.*, vol. 37, no. 4, pp. 667–676, Aug. 2015.

[23] 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.

[24] H. S. Stone, "Parallel processing with the perfect shuffle," *IEEE Trans. Comput.*, vol. COM-20, no. 2, pp. 153–161, Feb. 1971.

[25] M. Garrido and P. Pirsch, "Continuous-flow matrix transposition using memories," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 67, no. 9, pp. 3035–3046, Sep. 2020.

[26] 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*, early access, Feb. 16, 2023, doi: 10.1109/TCSII.2023.3245823.

[27] 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, Jan. 1965.

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

[29] M. Garrido, J. Grajal, and O. Gustafsson, "Optimum circuits for bit-dimension permutations," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 27, no. 5, pp. 1148–1160, May 2019.

[30] M. Garrido, "Efficient hardware architectures for the computation of the FFT and other related signal processing algorithms in real time," Ph.D. dissertation, Dept. Signals, Syst. Radiocommunications, Universidad Politécnica de Madrid, Madrid, Spain, Dec. 2009.

[31] M. Garrido, "A new representation of FFT algorithms using triangular matrices," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 63, no. 10, pp. 1737–1745, Oct. 2016.

[32] P. Paz and M. Garrido, "Efficient implementation of complex multipliers on FPGAs using DSP slices," *J. Signal Process. Syst.*, vol. 95, no. 4, pp. 543–550, Apr. 2023.

[33] Z.-G. Ma, X.-B. Yin, and F. Yu, "A novel memory-based FFT architecture for real-valued signals based on a radix-2 decimation-in-frequency algorithm," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 62, no. 9, pp. 876–880, Sep. 2015.

Zeynep Kaya received the M.Sc. and Ph.D. degrees in electrical and electronics engineering from Eskisehir Osmangazi University, Turkey, in 2015 and 2021, respectively.

Since 2021, she has been an Assistant Professor with Bilecik Şeyh Edebali University. Since September 2022, she has also been with Universidad Politénica de Madrid (UPM), Spain, as a Post-Doctoral Researcher. Her current research interests are optimized hardware architectures for the fast Fourier transform (FFT) including data management and memory addressing schemes. Her research interests include high-performance circuits, designs for small area, low latency, and low power consumption.

Mario Garrido (Senior Member, IEEE) received the M.Sc. and Ph.D. degrees in electrical engineering from Universidad Politénica de Madrid (UPM), Madrid, Spain, in 2004 and 2009, respectively.

In 2010, he moved to Sweden to work as a Post-Doctoral Researcher with the Department of Electrical Engineering, Linköping University, where he was an Associate Professor from 2012 to 2019. In 2019, he moved back to UPM, where he holds a Ramón y Cajal Research Fellowship. So far, he has been the author of more than 50 scientific publications. His research focuses on optimized hardware design for signal processing applications. This includes the design of hardware architectures for the fast Fourier transform (FFT), circuits for data management, the CORDIC algorithm, neural networks, and circuits to calculate statistical and mathematical operations. His research interests include high-performance circuits for real-time computation, and designs for small area and low power consumption. In 2022, he appeared in the "World's Top 2% Scientists List" elaborated by Stanford University.