Serial Butterflies for Non-Power-of-Two FFT Architectures in 5G and Beyond

Víctor Manuel Bautista, Mario Garrido, *Senior Member, IEEE*, and Marisa López-Vallejo, *Senior Member, IEEE*

***Abstract*— This paper presents new serial butterflies for non-power-of-two (NP2) fast Fourier transform (FFT) architectures. The paper considers radices 2, 3, 4, and 5, which are used in FFTs for 5G systems. Current designs for non-power-of-two FFTs are mostly based on the single-path delay feedback (SDF) architecture. This type of architecture processes data arriving in series. However, it uses butterflies with several parallel inputs. This results in low utilization, as the butterflies have to wait for all the inputs before they start to process them. Conversely, the proposed approach allows to calculate the butterflies on data that arrive in series. This removes waiting times and reduces the number of hardware components such as multipliers and adders. As a result, the proposed butterflies achieve high performance and provide a significant reduction in area and power consumption with respect to parallel butterflies. Thus, they are an efficient solution when data must be processed in series in the butterflies.**

***Index Terms*— Fast Fourier transform (FFT), non-power-of-two (NP2), pipeline architecture, serial butterfly, single-path delay feedback (SDF), serial commutator (SC).**

I. Introduction

THE Fourier transform [1] is a key component in 5G communication systems [2]. This mathematical operation transforms a signal from the time domain into the frequency domain. The discrete version of the Fourier transform is called discrete Fourier transform (DFT). To calculate the DFT, the fast Fourier transform (FFT) algorithm proposed by Cooley and Tukey [3] reduces the operation complexity from $\mathcal{O}(N^2)$ in the DFT to $\mathcal{O}(N \log N)$ in the FFT.

In 5G communications, the size of the FFTs is obtained as a product of powers of 2, 3, and 5, as is detailed in its physical layer description [2]. This motivates the need for designing non-power-of-two (NP2) FFTs. During the 20th century, several algorithms were proposed to make NP2 FFTs more efficient, such as those by Rader [4] and Winograd [5]. Furthermore, other NP2 algorithms such as the prime factor algorithm [6], [7], [8] have been proposed.

Manuscript received 22 March 2023; revised 1 July 2023; accepted 19 July 2023. Date of publication 2 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 and in part by MCIN/AEI/10.13039/501100011033 and "ESF-Investing in your future" under Grant RYC2018-025384-I. This article was recommended by Associate Editor J. Di. *(Corresponding author: Víctor Manuel Bautista.)*

The authors are with the Department of Electronic Engineering, ETSI de Telecomunicación, Universidad Politécnica de Madrid (UPM), 28040 Madrid, Spain (e-mail: victor.bautista@upm.es; mario.garrido@upm.es; m.lopez.vallejo@upm.es).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCSI.2023.3298207.

Digital Object Identifier 10.1109/TCSI.2023.3298207

When the FFT algorithm is implemented in hardware, pipelined architectures allow for high performance [9], [10], [11], [12], [13], [14], [15]. In fact, the field of pipelined FFT hardware architectures has been deeply developed during the last decades [16]. These designs have reached a high degree of optimization for power-of-two (P2) sizes [17]. Compared to them, the architectures that consider NP2 sizes have been barely explored, due to the higher complexity that algorithms for NP2 sizes [4], [5], [6], [7], [8] involve. The consequence of this fact for communication systems is that NP2 FFT architectures are barely used. Even when the most suitable FFT size in the system were a non-power-of-two FFT, it is common to use a higher size that is a power of two instead.

Nowadays, pipelined FFT hardware architectures for NP2 sizes mostly consider single-path delay feedback (SDF) architectures [18], [19], [20], [21], [22], [23], [24], [25], [26], with the exception of [27]. However, for NP2 sizes, SDF architectures are not as efficient as could be expected: Although SDF architectures process data in series at a rate of one sample per clock cycle, the butterflies that they use operate data in parallel. This means that the butterflies are only working for a fraction of the time, whereas the rest of the time they wait for new data. This leads to a utilization of the butterflies in SDF FFTs of $1/r$, where r is the radix of the butterfly. Thus, in the best case, the utilization is 50% when using radix-2 butterflies, 33% for radix-3, 25% for radix-4, and 20% for radix-5. As a consequence, there is room for improving the butterflies by increasing their utilization and removing waiting times. In order to achieve this, a feasible approach is to develop serial butterflies with one input and one output that process one sample per clock cycle, instead of processing several samples per clock cycle in parallel. With this aim, previous works have been proposed in [26], [27], and [28]. In [26], a novel design for a radix-3 SDF butterfly is presented. This butterfly distributes the operations along three stages connected in series. In [27] a 2-parallel radix-3 butterfly is designed, which processes two simultaneous 3-point FFTs by sharing adders and multipliers along a pipeline. Finally, in [28], radix-3 and radix-5 serial butterflies are designed by using a low number of adders and registers. These butterflies are based on reusing radix-2 modules.

In this paper, new efficient serial pipelined butterflies for radices 2, 3, 4, and 5 are proposed. These butterflies reach a high utilization that allows for achieving low area and high performance simultaneously. The proposed designs focus on minimizing hardware-consuming components such as adders and multipliers. The strategy that has been followed is to

divide the complex-valued calculations of the butterflies into operations with real-valued data. Then, these operations are distributed along a pipelined circuit. This, combined with a carefully designed data management, leads to serial butterflies with a high degree of optimization. The preliminary version of the serial butterflies proposed in this paper was developed in the author's Bachelor Thesis [29]. This paper provides the scientific publication of that work and completes it with new implementations, experimental results, and comparison. The proposed butterflies are suitable for future non-power-of-two serial commutator (SC) FFT architectures [10], where the processing elements operate on data that arrive in series in consecutive clock cycles.

The novelty and contribution of the paper can be observed at various levels. First, this paper is the first work that deals in a rigorous way with the design of serial butterflies. Second, the paper is the first one that highlights and faces one of the key problems in NP2 FFTs, which is the low utilization of butterflies. Third, the paper presents efficient solutions to tackle this problem. Fourth, the challenge of designing the butterflies in the paper required a thorough analysis of the data flow in order to obtain an order of operations that reduces the hardware components. Finally, we have pursued that the paper is complete, providing any information related to the proposed butterflies that may be relevant for the reader. The reason is that the design of optimized butterflies is fundamental for the design of efficient NP2 FFTs. Without them, future NP2 FFT will not be feasible in communication systems, because they would still require a large amount of hardware, as they do nowadays. Thus, the final goal that this work pursues is to develop NP2 FFT architectures that are as efficient as power-of-two ones. With this goal, communication systems will be able to implement NP2 FFTs instead of being forced to resort to P2 FFT sizes. This ambitious goal of deriving efficient NP2 FFT architectures will take place in several steps. In this paper, we set the first stone to build NP2 FFT architecture by developing efficient butterflies for NP2. In future works, we will present new efficient algorithms for NP2 FFTs, shuffling circuits to calculate the permutations in NP2 FFTs, and, finally, the desired efficient NP2 FFT architectures.

The paper is organized as follows: In Section II, the state-of-the-art is reviewed. In Section III, the proposed serial butterflies are presented and analyzed in detail. In Section IV, the proposed designs are compared to previous ones. In Section V, implementation results on FPGA and ASIC are reported and compared with parallel butterflies. Finally, in Section VI, the main conclusions of the paper are provided.

II. Background

A. The FFT

An N-point discrete Fourier transform (DFT) of a discrete complex signal $x[n]$ is defined as

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

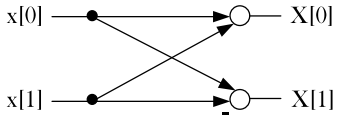

Fig. 1. Signal flow graph of the radix-2 butterfly.

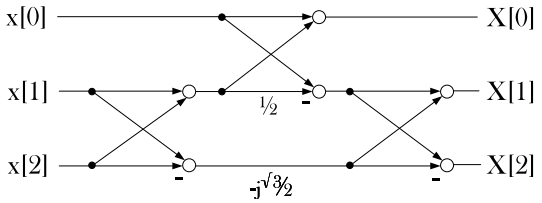

Fig. 2. Signal flow graph of the radix-3 butterfly.

where $X[k]$ represents the output at frequency k. The term $W_N^{nk} = e^{-j\frac{2\pi}{N}nk}$ is called twiddle factor and calculates a rotation in the complex plane. The FFT algorithm divides the N-point DFT into smaller DFTs whose sizes are factors of N, being the product of all of these sizes equal to N. The minimum possible sizes correspond to the case when N is decomposed into prime numbers. The processing elements that calculate these small DFTs are called butterflies. In this paper, we consider butterflies of sizes 2, 3, 4, and 5, which are relevant sizes in FFTs for 5G.

B. Butterflies

A radix-r butterfly calculates an r-point DFT. Fig. 1 shows the signal flow graph (SFG) of a radix-2 butterfly. It consists of an addition and a subtraction according to

$$X[0] = x[0] + x[1], \quad (2a)$$
$$X[1] = x[0] - x[1]. \quad (2b)$$

It can be observed that these operations correspond to the calculation of the DFT in (1) for $N = 2$ points.

Fig. 2 shows the signal flow graph of a radix-3 butterfly based on Rader's algorithm [4]. According to (1), it carries out rotations by 0°, 120° and −120°. The operations that are extracted from the flow graph are

$$X[0] = x[0] + x[1] + x[2], \quad (3a)$$
$$X[1] = x[0] - \frac{1}{2}(x[1] + x[2]) - j\frac{\sqrt{3}}{2}(x[1] - x[2]), \quad (3b)$$
$$X[2] = x[0] - \frac{1}{2}(x[1] + x[2]) + j\frac{\sqrt{3}}{2}(x[1] - x[2]). \quad (3c)$$

The radix-3 flow graph reuses the products that appear in (3b) and (3c). Thus, only two multiplications have to be calculated in the flow graph. The multiplication by $-j\frac{\sqrt{3}}{2}$ involves two real multiplications, whereas the multiplication by $1/2$ can be calculated with a bit shift and, therefore, it does not have any hardware cost. Additionally, in the radix-3 butterfly, 6 complex additions are calculated.

Fig. 3 represents the flow graph of the radix-4 butterfly. The operations that are carried out are

$$X[0] = (x[0] + x[2]) + (x[1] + x[3]), \quad (4a)$$

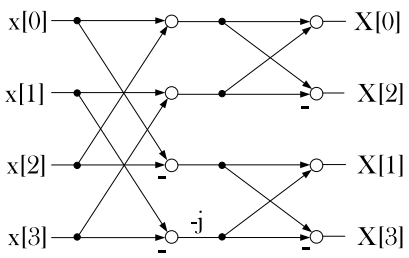

Fig. 3. Signal flow graph of the radix-4 butterfly.

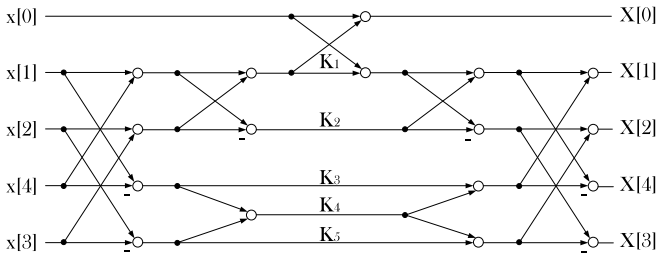

Fig. 4. Signal flow graph of the radix-5 butterfly.

TABLE I
VALUES FOR THE COEFFICIENTS IN THE RADIX-5 FLOW GRAPH OF FIG. 4

Coeff	Equation	Value
K_1	$-\frac{1}{4}$	-0.2500
K_2	$\frac{1}{2}\left(\cos\left(\frac{2\pi}{5}\right)-\cos\left(\frac{4\pi}{5}\right)\right)$	0.5590
K_3	$j\left(\sin\left(\frac{4\pi}{5}\right)-\sin\left(\frac{2\pi}{5}\right)\right)$	$-j\cdot 0.3633$
K_4	$-j\sin\left(\frac{4\pi}{5}\right)$	$-j\cdot 0.5878$
K_5	$j\left(\sin\left(\frac{4\pi}{5}\right)+\sin\left(\frac{2\pi}{5}\right)\right)$	$j\cdot 1.5388$

$$X[1] = (x[0] - x[2]) - j(x[1] - x[3]), \tag{4b}$$
$$X[2] = (x[0] + x[2]) - (x[1] + x[3]), \tag{4c}$$
$$X[3] = (x[0] - x[2]) + j(x[1] - x[3]). \tag{4d}$$

The flow graph of the radix-4 butterfly includes a multiplication by $-j$, which corresponds to a rotation of $-90°$ in the complex plane. Rotations by 0°, 90°, 180, and $-90°$ are called trivial rotations because they can be calculated by changing the real/imaginary parts of the input and/or changing its sign. This makes it possible to avoid the implementation of multipliers in the radix-4 butterfly. Regarding adders, the radix-4 butterfly requires 8 complex adders.

Fig. 4 shows a signal flow graph of the radix-5 butterfly. It is based on the Winograd's algorithm [5]. However, in Fig. 4 we have reordered the operations of the third stage so that the first multiplication is by $K_1 = -1/4$. This allows to replace the multiplier in the Winograd's algorithm with a bit shift in hardware. Table I lists the values of the coefficients for the multiplications. The values of K_3 and K_5 also change with respect to the Winograd's algorithm, because a different order for the input data is considered. The flow graph in Fig. 4 requires the calculation of 17 complex additions and 8 real multiplications.

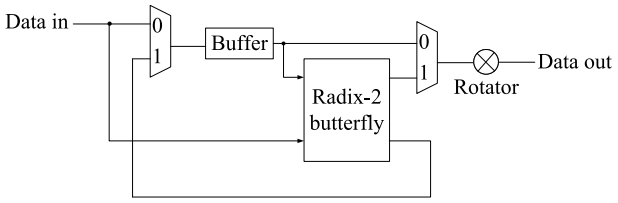

Fig. 5. Stage of a radix-2 SDF FFT.

C. SDF FFT Architectures

SDF FFT architectures are the most common pipelined architectures used to process NP2 FFTs [19], [21], [22], [23], [24], [25], [26], [27]. Fig. 5 shows an SDF stage that uses a radix-2 butterfly. Input data arrive in series during consecutive clock cycles. The first half of the inputs is streamed to the buffer. While the buffer is being filled, the butterfly is not used. When the buffer is full, the output of the buffer is streamed to the upper input of the butterfly to operate these samples with the new input data. When the butterfly starts to work, half of the processed data is streamed to the rotator, while the other half is stored in the buffer. Finally, the data stored in the buffer is streamed to the output. This process repeats periodically as new data arrive at the circuit.

In the general case of radix-r, the stage consists of $r-1$ buffers, a radix-r butterfly, and multiplexers. The higher the radix, the more buffers the circuit has and the less time the butterfly is used. As a result, the utilization of each butterfly in an SDF architecture is reduced to $1/r$. Thus, the butterflies reach 50% utilization in radix-2, 33% in radix-3, 25% in radix-4, and 20% in radix-5.

III. PROPOSED SERIAL BUTTERFLIES

A. Theoretical Limits

Radix-r butterflies in SDF architectures have r inputs and they correspond to the direct implementation of the flow graphs in Figs. 1, 2, 3 and 4. However, as serial FFT architectures only process one input per clock cycle, there is no need to use butterflies with several inputs in parallel. Thus, the proposed designs have only one input and one output, and process one sample per clock cycle. As a result, it is possible to reduce the area of the butterflies by serializing the operations of the signal flow graphs. The minimum number of real adders and real multipliers that a serial implementation of a butterfly can reach are

$$\text{Real adders}_{\min} = \left\lceil \frac{\text{Real additions in SFG}}{r} \right\rceil, \tag{5a}$$
$$\text{Real multipliers}_{\min} = \left\lceil \frac{\text{Real multiplications in SFG}}{r} \right\rceil, \tag{5b}$$

where $\lceil\cdot\rceil$ represents a ceiling operation. Table II shows the number of real operations that appear in the direct implementation of the signal flow graph and the minimum number of real multipliers and real adders that a serial implementation can reach. Therefore, it is theoretically possible to reduce the number of elements by a factor r or close to r, leading to less area usage.

TABLE II
THEORETICAL MINIMUM NUMBER OF ELEMENTS THAT CAN BE ACHIEVED IN A SERIAL IMPLEMENTATION OF A BUTTERFLY

	Parallel		Serial	
r	Real Additions	Real Multiplications	Minimum Real Adders	Minimum Real Multipliers
2	4	0	2	0
3	12	2	4	1
4	16	0	4	0
5	34	8	7	2

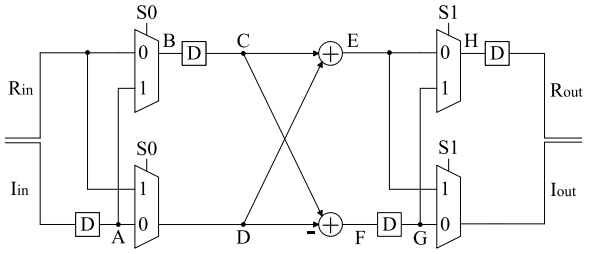

Fig. 6. Proposed radix-2 serial butterfly.

B. Proposed Radix-2 Serial Butterfly

In the radix-2 butterfly in Fig. 1, data are complex-valued. Thus, it calculates

$$X_{r,0} = x_{r,0} + x_{r,1}, \tag{6a}$$
$$X_{r,1} = x_{r,0} - x_{r,1}, \tag{6b}$$
$$X_{i,0} = x_{i,0} + x_{i,1}, \tag{6c}$$
$$X_{i,1} = x_{i,0} - x_{i,1}, \tag{6d}$$

where $x[0] = x_{r,0} + jx_{i,0}$ is the upper input, $x[1] = x_{r,1} + jx_{i,1}$ is the lower input, $X[0] = X_{r,0} + jX_{i,0}$ is the upper output and $X[1] = X_{r,1} + jX_{i,1}$ is the lower output. According to this, Fig. 6 shows the proposed radix-2 serial butterfly. The design of this butterfly is inspired by the serial commutator processing element [10] and consists of an adder, a subtractor, four multiplexers, four registers, and zero multipliers. Note that the number of real adders and real multipliers correspond to the minimum values according to Table II. The circuit processes one sample per clock cycle, which is first separated into its real and imaginary parts. These parts are sent to the upper and lower branches of the circuit, respectively. Before and after the adders, the circuit includes serial-parallel permutation circuits [30], which consist of two multiplexers and two registers each. These circuits are used for reordering data. Finally, the circuit provides the real and imaginary parts of the data at the upper and lower branches, respectively.

Table III shows the timing diagram of the proposed radix-2 serial butterfly. Each row of the timing diagram corresponds to a signal of the circuit shown in Fig. 6. Note that letters are added to Fig. 6 to identify these signals. The first two signals in Table III represent the values of the control signals of the multiplexers. The next two rows represent the real and imaginary parts of the input data, which arrive at the same clock cycle. Signals A to H represent intermediate nodes of the circuit. Finally, the last two rows represent the real and imaginary parts of the output data, which are provided at the same clock cycle. Pairs of data to be processed in the butterfly arrive in consecutive clock cycles. Thus, the serial-parallel permutation circuit permutes data so that the real part of the second sample and the imaginary part of the first sample are exchanged at C and D. This permutation makes it possible to operate the real parts of the data first and the imaginary parts during the next clock cycle, according to the set of equations (6). Then, the butterfly provides the real and imaginary parts to the output at the same clock cycle by using the second serial-parallel permutation circuit. As each serial-parallel permutation circuit has a latency of one clock cycle, the butterfly has a total latency of two clock cycles.

TABLE III
TIMING DIAGRAM OF THE PROPOSED RADIX-2 SERIAL BUTTERFLY IN FIG. 6

	Time			
Signal	0	1	2	3
S0	0	1	0	1
S1	1	0	1	0
R_{in}	$x_{r,0}$	$x_{r,1}$		
I_{in}	$x_{i,0}$	$x_{i,1}$		
A		$x_{i,0}$	$x_{i,1}$	
B	$x_{r,0}$	$x_{i,0}$		
C		$x_{r,0}$	$x_{i,0}$	
D		$x_{r,1}$	$x_{i,1}$	
E		$X_{r,0}$	$X_{i,0}$	
F		$X_{r,1}$	$X_{i,1}$	
G			$X_{r,1}$	$X_{i,1}$
H		$X_{r,0}$	$X_{r,1}$	
R_{out}			$X_{r,0}$	$X_{r,1}$
I_{out}			$X_{i,0}$	$X_{i,1}$

C. Proposed Radix-3 Serial Butterfly

Fig. 7 shows the implementation of the proposed radix-3 serial butterfly. As in its flow graph in Fig. 2, the proposed hardware implementation distributes the required operations along three stages. The dashed lines placed after the adders and multipliers represent pipeline registers used during the implementation to improve the maximum clock frequency. The number in the upper side of some dash lines indicates the number of pipeline registers connected in series and dash lines with no number represent a single pipeline register. The proposed circuit reaches the minimum number of real multipliers according to Table II: The multiplication by $1/2$ is implemented by a bit-shift, which does not have any hardware cost, and only one real multiplier is used in the proposed radix-3 serial butterfly. Fig. 8 shows the implementation of the real multiplier using shift-and-add operations. The multiplication by $\frac{\sqrt{3}}{2}$ is approximated by

$$\frac{\sqrt{3}}{2} \approx \frac{887}{1024} = \frac{((8-1)\cdot 16 - 1)\cdot 8 - 1}{1024} = 0.8662. \tag{7}$$

As a result, the proposed radix-3 serial butterfly uses 6 real adders plus a real multiplier that is implemented with 3 real adders, leading to a total of 9 real adders. The circuit also includes 12 multiplexers and 8 registers.

Table IV shows the timing diagram of the proposed circuit. Note that input data arrive in natural order as $x[0]$, $x[1]$, $x[2]$ in consecutive clock cycles. The intermediate calculations are detailed in Table V. It can be observed that certain signals do not need to be operated in the adders in Table V. For them,

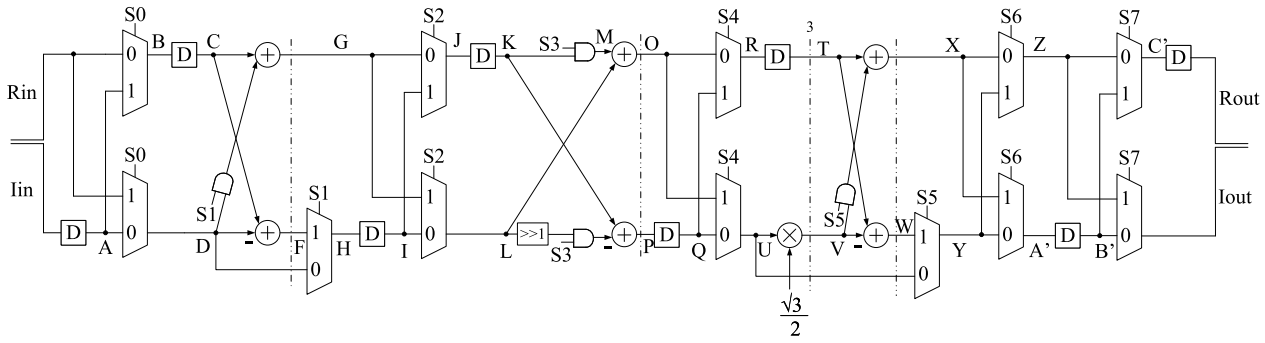

Fig. 7. Proposed radix-3 serial butterfly.

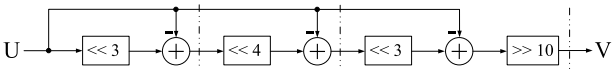

Fig. 8. Shift-and-add multiplier by 887/1024 for the radix-3 serial butterfly in Fig. 7.

TABLE IV
TIMING DIAGRAM OF THE PROPOSED RADIX-3 SERIAL BUTTERFLY IN FIG. 7

	Time						
Signal	0	1	2	3	4	5	6
S0	0	0	1	0	0	1	0
S1	1	0	1	1	0	1	1
S2	1	0	1	1	0	1	1
S3	1	0	1	1	0	1	1
S4	1	1	0	1	1	0	1
S5	0	1	1	0	1	1	0
S6	0	0	1	0	0	1	0
S7	0	0	1	0	0	1	0
R_{in}	$x_{r,0}$	$x_{r,1}$	$x_{r,2}$				
I_{in}	$x_{i,0}$	$x_{i,1}$	$x_{i,2}$				
C		$x_{r,0}$	$x_{r,1}$	$x_{i,1}$			
D		$x_{i,0}$	$x_{r,2}$	$x_{i,2}$			
G		$x_{r,0}$	α	γ			
H		$x_{i,0}$	β	δ			
K			$x_{r,0}$	$x_{i,0}$	β		
L			α	γ	δ		
O			$X_{r,0}$	$X_{i,0}$	δ		
P			ϵ	ζ	β		
T				$X_{r,0}$	ϵ	ζ	
U				$X_{i,0}$	δ	β	
X				$X_{r,0}$	$X_{r,1}$	$X_{i,2}$	
Y				$X_{i,0}$	$X_{r,2}$	$X_{i,1}$	
R_{out}					$X_{r,0}$	$X_{r,1}$	$X_{r,2}$
I_{out}					$X_{i,0}$	$X_{i,1}$	$X_{i,2}$

TABLE V
CALCULATIONS IN THE PROPOSED RADIX-3 SERIAL BUTTERFLY

First Stage	Second Stage	Third Stage
$\alpha = x_{r,1} + x_{r,2}$	$\epsilon = x_{r,0} - \frac{\alpha}{2}$	$X_{r,1} = \epsilon + \frac{\sqrt{3}}{2}\delta$
$\beta = x_{r,1} - x_{r,2}$	$\zeta = x_{i,0} - \frac{\gamma}{2}$	$X_{i,1} = \zeta - \frac{\sqrt{3}}{2}\beta$
$\gamma = x_{i,1} + x_{i,2}$	$X_{r,0} = x_{r,0} + \alpha$	$X_{r,2} = \epsilon - \frac{\sqrt{3}}{2}\delta$
$\delta = x_{i,1} - x_{i,2}$	$X_{i,0} = x_{i,0} + \gamma$	$X_{i,2} = \zeta + \frac{\sqrt{3}}{2}\beta$

the circuit includes logic gates and multiplexers that are used to bypass the adders. As for the input data, the outputs are also provided in natural order. By considering the delays of the permutation circuits, the proposed radix-3 serial butterfly has a latency of four clock cycles.

TABLE VI
TIMING DIAGRAM OF THE PROPOSED RADIX-4 BUTTERFLY IN FIG. 9

	Time									
Sig.	0	1	2	3	4	5	6	7	8	9
S0	0	1	0	1	0	1	0	1	0	1
S1	0	0	1	1	0	0	1	1	0	0
S2	1	0	0	0	1	0	0	0	1	0
S3	1	0	1	1	1	0	1	1	1	0
S4	0	1	0	0	0	1	0	0	0	1
S5	1	1	0	0	1	1	0	0	1	1
S6	0	0	1	1	0	0	1	1	0	0
S7	1	0	1	0	1	0	1	0	1	0
R_{in}	$x_{r,0}$	$x_{r,1}$	$x_{r,2}$	$x_{r,3}$						
I_{in}	$x_{i,0}$	$x_{i,1}$	$x_{i,2}$	$x_{i,3}$						
H			$x_{r,0}$	$x_{i,0}$	$x_{r,3}$	$x_{i,1}$				
I			$x_{r,2}$	$x_{i,2}$	$x_{r,1}$	$x_{i,3}$				
J			α	β	γ	δ				
K			ϵ	ζ	η	θ				
Q					α	β	ϵ	ζ		
R					γ	δ	θ	η		
T					$X_{r,0}$	$X_{i,0}$	$X_{r,1}$	$X_{i,1}$		
U					$X_{r,2}$	$X_{i,2}$	$X_{r,3}$	$X_{i,3}$		
R_{out}							$X_{r,0}$	$X_{r,1}$	$X_{r,2}$	$X_{r,3}$
I_{out}							$X_{i,0}$	$X_{i,1}$	$X_{i,2}$	$X_{i,3}$

TABLE VII
CALCULATIONS IN THE PROPOSED RADIX-4 SERIAL BUTTERFLY

First Stage	Second Stage
$\alpha = x_{r,0} + x_{r,2}$	$X_{r,0} = \alpha + \gamma$
$\beta = x_{i,0} + x_{i,2}$	$X_{i,0} = \beta + \delta$
$\gamma = x_{r,3} + x_{r,1}$	$X_{r,1} = \epsilon + \theta$
$\delta = x_{i,1} + x_{i,3}$	$X_{i,1} = \zeta + \eta$
$\epsilon = x_{r,0} - x_{r,2}$	$X_{r,2} = \alpha - \gamma$
$\zeta = x_{i,0} - x_{i,2}$	$X_{i,2} = \beta - \delta$
$\eta = x_{r,3} - x_{r,1}$	$X_{r,3} = \epsilon - \theta$
$\theta = x_{i,1} - x_{i,3}$	$X_{i,3} = \zeta - \eta$

D. Proposed Radix-4 Serial Butterfly

The operations required to process a 4-point FFT are described in the flow graph of Fig. 3. There are two clearly distinguished stages, which consist of four complex additions each. Based on it, Fig. 9 shows the proposed radix-4 butterfly. As in its flow graph, the proposed hardware implementation distributes the required operations along two stages. These stages include four real adders and zero multipliers, which correspond to the minimum values according to Table II.

Table VI shows the timing diagram of the circuit and the operations are detailed in Table VII. Input data arrive in natural order as $x[0]$, $x[1]$, $x[2]$, $x[3]$. As the first sample is

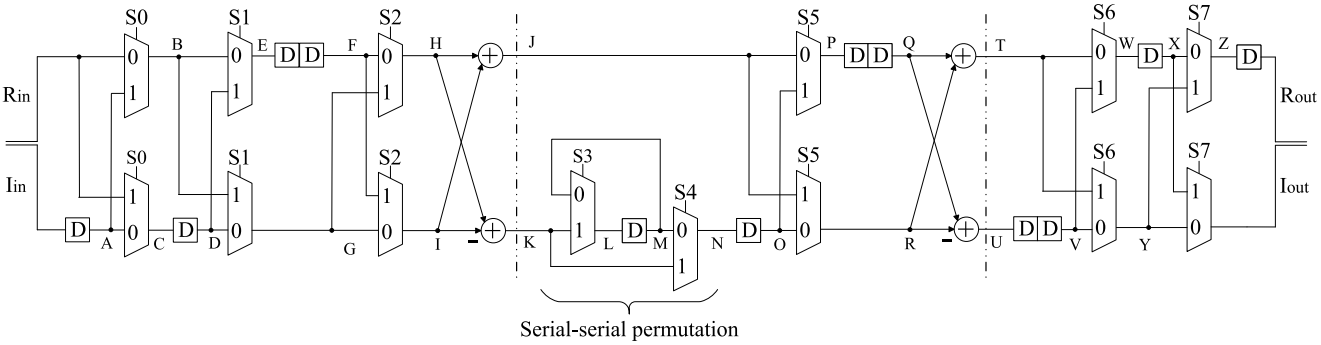

Fig. 9. Proposed radix-4 serial butterfly.

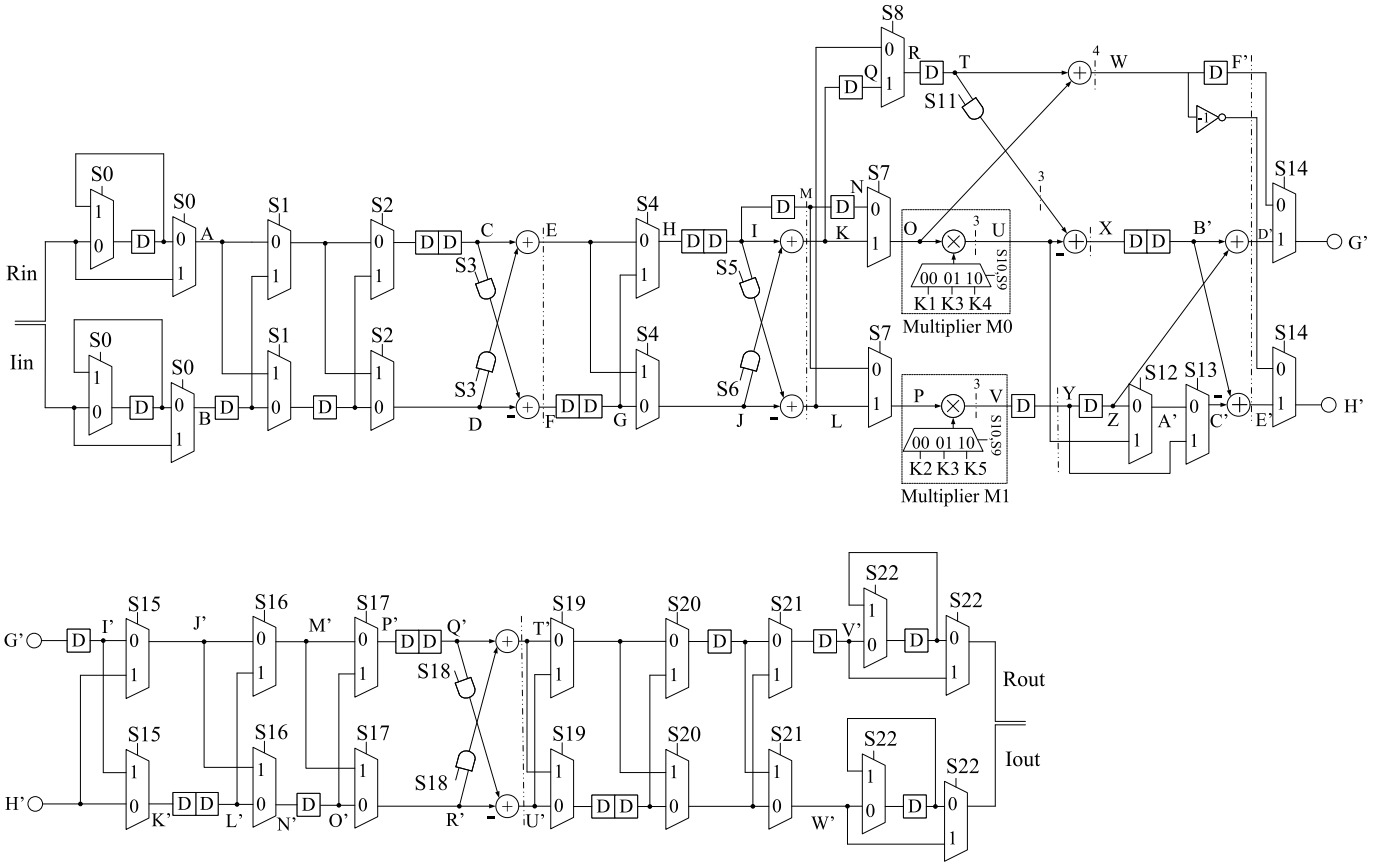

Fig. 10. Proposed radix-5 serial butterfly.

operated with the third one, and the second sample is operated with the fourth one, a serial-parallel permutation circuit is included at the input of the circuit. This circuit has dual functionality. First, it permutes the imaginary parts of $x[0]$ and $x[2]$ with the real parts of $x[1]$ and $x[3]$. Then, it places pairs of inputs to be operated together in the same clock cycle, as can be seen in signals H and I in Table VI. After the operations of the first stage, the proposed serial butterfly uses a serial-serial permutation circuit [30] to exchange data that arrive in consecutive clock cycles through the lower path. Finally, the output is provided in natural order by using an additional serial-parallel permutation circuit. As a result, considering the delays of the permutation circuits, the butterfly has a latency of 6 clock cycles.

E. Proposed Radix-5 Serial Butterfly

Fig. 10 shows the proposed radix-5 serial butterfly. The aim of this implementation is to use the minimum possible number of real multipliers, as well as a number of real adders in line with the number of stages in the flow graph of Fig. 4. As in the flow graph, the proposed hardware implementation distributes the required operations along five stages. These stages include 10 real adders and two real multipliers, which means that the minimum number of real multipliers according to Table II is achieved. Table VIII shows the operations that are calculated at each stage. The multiplier constants of the radix-5 serial butterfly in Fig. 10, which also appear in Table VIII, are the ones listed in the first and second columns of Table IX. Note that the magnitude of these constants is the same as the

TABLE VIII
CALCULATIONS IN THE PROPOSED RADIX-5 SERIAL BUTTERFLY

First Stage	Second Stage	Third Stage	Fourth Stage	Fifth stage
$\epsilon = x_{r,1} + x_{r,4}$	$\alpha^{'} = \alpha + \beta$	$\lambda = x_{r,0} + K_1\alpha^{'}$	$\alpha^{''} = \lambda + K_2\beta^{'}$	$X_{r,1} = \alpha^{''} - \beta^{''}$
$\zeta = x_{r,2} + x_{r,3}$	$\beta^{'} = \alpha - \beta$	$\mu = x_{i,0} + K_1\gamma^{'}$	$\beta^{''} = K_4\zeta^{'} - K_3\eta$	$X_{i,1} = \gamma^{''} + \delta^{''}$
$\eta = x_{i,1} + x_{i,4}$	$\gamma^{'} = \gamma + \delta$	$X_{r,0} = x_{r,0} + \alpha^{'}$	$\gamma^{''} = \mu + K_2\delta^{'}$	$X_{r,2} = \epsilon^{''} - \zeta^{''}$
$\theta = x_{i,2} + x_{i,3}$	$\delta^{'} = \gamma - \delta$	$X_{i,0} = x_{i,0} + \gamma^{'}$	$\delta^{''} = K_4\epsilon^{'} - K_3\epsilon$	$X_{i,2} = \eta^{''} + \theta^{''}$
$\alpha = x_{r,1} - x_{r,4}$	$\epsilon^{'} = \epsilon + \zeta$		$\epsilon^{''} = \lambda - K_2\beta^{'}$	$X_{r,3} = \epsilon^{''} + \zeta^{''}$
$\beta = x_{r,2} - x_{r,3}$	$\zeta^{'} = \eta + \theta$		$\zeta^{''} = K_4\zeta^{'} + K_5\theta$	$X_{i,3} = \eta^{''} - \theta^{''}$
$\gamma = x_{i,1} - x_{i,4}$			$\eta^{''} = \mu - K_2\delta^{'}$	$X_{r,4} = \alpha^{''} + \beta^{''}$
$\delta = x_{i,2} - x_{i,3}$			$\theta^{''} = K_4\epsilon^{'} + K_5\zeta$	$X_{i,4} = \gamma^{''} - \delta^{''}$

TABLE IX
VALUES FOR THE CONSTANTS OF THE PROPOSED RADIX-5 SERIAL BUTTERFLY IN FIG. 10

Constant	Value	Approximated Value
Multiplier M0		
$K1$	-0.2500	$-128/512 = -0.2500$
$K4$	-0.5878	$-301/512 = -0.5879$
$K3$	0.3633	$186/512 = 0.3633$
Multiplier M1		
$K2$	0.5590	$143/256 = 0.5586$
$K5$	-1.5388	$-394/256 = -1.5391$
$K3$	0.3633	$93/256 = 0.3633$

magnitude reported in Table I for the radix-5 parallel butterfly. However, their phase is different in some cases. Additionally, both real multipliers have been implemented with shift-and-add operations as reconfigurable multiple constant multipliers (RMCM) [31]. The upper multiplier, M0, is shown in Fig. 11 and the lower multiplier, M1, is shown in Fig. 12. The multiplier M1 has been designed with the heuristics in [31]. The control signals S9 and S10 that appear in these circuits are the same control signals that appear in Fig. 10. The third column of Table IX shows the approximated values used in the shift-and-add circuits. Both shift-and-add circuits use 3 real adders, which are shared for every constant with the help of additional multiplexers. As a result, the proposed serial radix-5 butterfly uses 10 real adders and two real multipliers implemented with three real adders each.

Table X shows the timing diagram of the circuit. As in all the proposed serial butterflies, inputs arrive in natural order and a circuit that reorders the inputs is needed. This circuit consists of serial-serial permutation circuits linked to a serial-parallel permutation circuit. The serial-serial permutation circuits exchange data from natural order to $x[0]$, $x[1]$, $x[2]$, $x[4]$, $x[3]$ at A and B. Then, the serial-parallel permutation circuit places pairs of inputs to be operated together at C and D in the same clock cycle. Note that one additional parallel branch has been added to the circuit at the third stage due to the fact that the flow graph of radix-5 has more than five branches at stage 3. The real and imaginary parts of $x[0]$ are bypassed using logic gates during the first two stages and sent to this additional branch in consecutive clock cycles, as can be seen in signal T. Once they are operated,

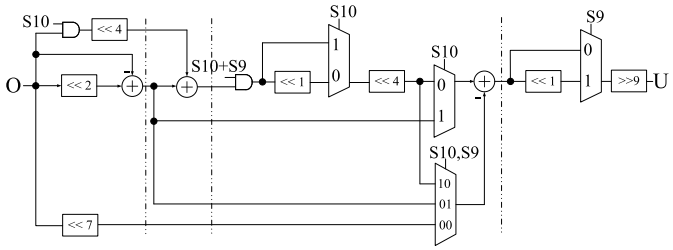

Fig. 11. Shift-and-add multiplier M0 for the radix-5 serial butterfly in Fig. 10.

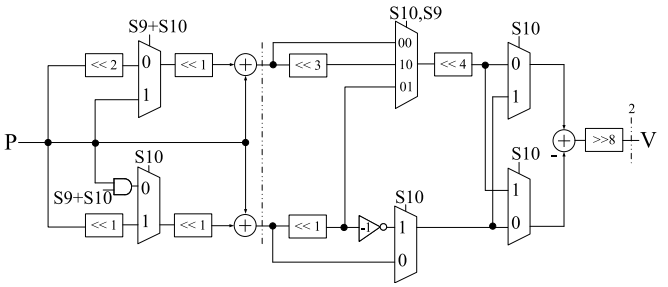

Fig. 12. Shift-and-add multiplier M1 for the radix-5 serial butterfly in Fig. 10.

there is no need to maintain an additional branch and the first frequency, $X[0]$, returns to the regular path at G' and H'. After the next two stages, the output is provided in natural order by using an additional serial-parallel permutation circuit linked to serial-serial permutation circuits, whose functionality is dual to the input circuits. As a result, considering the delays of the datapath, the proposed serial radix-5 butterfly has a latency of 13 clock cycles.

IV. COMPARISON

Table XI shows the comparison between the proposed serial butterflies and previous approaches. Previous works include radix-3 serial butterflies [26], [28], a 2-parallel radix-3 butterfly [27] and a radix-5 serial butterfly [28].

The table compares the works in terms of real multipliers, real adders, real multiplexers, registers, throughput in samples per clock cycle, and latency in clock cycles (cyc.). For the number of real multipliers it is assumed that a complex multiplication uses four real multipliers and two real adders, and a complex multiplication by either a pure complex or a pure real constant requires two real multipliers.

The proposed radix-2 serial butterfly only requires 2 real adders, 4 real multiplexers, 4 registers, and no real multiplier.

TABLE X

TIMING DIAGRAM OF THE PROPOSED RADIX-5 SERIAL BUTTERFLY IN FIG. 10

	Time																	
Signal	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
S0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0
S1	1	0	0	1	0	1	0	0	1	0	1	0	0	1	0	1	0	0
S2	1	0	0	0	1	1	0	0	0	1	1	0	0	0	1	1	0	0
S3	1	1	1	0	1	1	1	1	0	1	1	1	1	0	1	1	1	1
S4	0	1	1	0	0	0	1	1	0	0	0	1	1	0	0	0	1	1
S5	0	1	1	0	0	0	1	1	0	0	0	1	1	0	0	0	1	1
S6	0	1	1	1	1	0	1	1	1	1	0	1	1	1	1	0	1	1
S7	0	1	1	1	1	0	1	1	1	1	0	1	1	1	1	0	1	1
S8	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0
S9	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0
S10	0	0	0	1	1	0	0	0	1	1	0	0	0	1	1	0	0	0
S11	0	1	1	0	0	0	1	1	0	0	0	1	1	0	0	0	1	1
S12	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0
S13	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0
S14	1	1	0	1	1	1	1	0	1	1	1	1	0	1	1	1	1	0
S15	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0
S16	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1
S17	0	1	1	0	0	0	1	1	0	0	0	1	1	0	0	0	1	1
S18	0	1	1	1	1	0	1	1	1	1	0	1	1	1	1	0	1	1
S19	0	1	0	1	0	0	1	0	1	0	0	1	0	1	0	0	1	0
S20	0	0	0	1	1	0	0	0	1	1	0	0	0	1	1	0	0	0
S21	1	0	0	1	0	1	0	0	1	0	1	0	0	1	0	1	0	0
S22	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0
R_{in}	$x_{r,0}$	$x_{r,1}$	$x_{r,2}$	$x_{r,3}$	$x_{r,4}$													
I_{in}	$x_{i,0}$	$x_{i,1}$	$x_{i,2}$	$x_{i,3}$	$x_{i,4}$													
A		$x_{r,0}$	$x_{r,1}$	$x_{r,2}$	$x_{r,4}$	$x_{r,3}$												
B		$x_{i,0}$	$x_{i,1}$	$x_{i,2}$	$x_{i,4}$	$x_{i,3}$												
C				$x_{r,0}$	$x_{r,1}$	$x_{i,1}$	$x_{r,2}$	$x_{i,2}$										
D				$x_{i,0}$	$x_{r,4}$	$x_{i,4}$	$x_{r,3}$	$x_{i,3}$										
E				$x_{r,0}$	α	γ	β	δ										
F				$-x_{i,0}$	ϵ	η	ζ	θ										
I						$x_{r,0}$	α	γ	ϵ	η								
J						$-x_{i,0}$	β	δ	ζ	θ								
K						$x_{r,0}$	α'	γ'	ϵ'	ζ'								
L						$x_{i,0}$	β'	δ'	$-\zeta$	$-\theta$								
O							α'	γ'	ϵ'	ζ'	ϵ							
P							β'	δ'	$-\zeta$	$-\theta$	η							
T							$x_{r,0}$	$x_{i,0}$	-	-	-							
U							$K_1\alpha'$	$K_1\gamma'$	$K_4\epsilon'$	$K_4\zeta'$	$K_3\epsilon$							
V							$K_2\beta'$	$K_2\delta'$	$K_5\zeta'$	$K_5\theta'$	$K_3\eta$							
W							$X_{r,0}$	$X_{i,0}$	-	-	-							
X							λ	μ	$K_4\epsilon'$	$K_4\zeta'$	$K_3\epsilon$							
B'									λ	μ	$K_4\epsilon'$	$K_4\zeta'$	-					
C'									$K_2\beta'$	$K_2\delta'$	$K_3\epsilon$	$K_3\eta$						
D'									α''	γ''	θ''	ζ''						
E'									ϵ''	η''	δ''	β''						
G'								$X_{r,0}$	α''	γ''	θ''	ζ''						
H'								$-X_{i,0}$	ϵ''	η''	δ''	β''						
Q'											$X_{r,0}$	α''	γ''	ϵ''	η''			
R'											$-X_{i,0}$	β''	δ''	ζ''	θ''			
T'											$X_{r,0}$	$X_{r,4}$	$X_{i,1}$	$X_{r,3}$	$X_{i,2}$			
U'											$X_{i,0}$	$X_{r,1}$	$X_{i,4}$	$X_{r,2}$	$X_{i,3}$			
V'													$X_{r,0}$	$X_{r,1}$	$X_{i,2}$	$X_{r,4}$	$X_{i,3}$	
W'													$X_{i,0}$	$X_{r,1}$	$X_{i,2}$	$X_{r,4}$	$X_{i,3}$	
R_{out}														$X_{r,0}$	$X_{r,1}$	$X_{r,2}$	$X_{r,3}$	$X_{r,4}$
I_{out}														$X_{i,0}$	$X_{i,1}$	$X_{i,2}$	$X_{i,3}$	$X_{i,4}$

TABLE XI

COMPARISON OF BUTTERFLIES IN TERMS OF HARDWARE COMPONENTS

	Radix-2	Radix-3				Radix-4	Radix-5	
Parameter	Proposed	[26]	[27]	[28]	Proposed	Proposed	[28]	Proposed
Real multipliers	0	0	0	2	0	0	4	0
Real adders	2	18	18	4	9	4	10	16
Real multiplexers	4	14	26	14	12	14	46	45
Real Registers	4	6	12	6	8	12	12	31
Throughput (samples/cyc.)	1	1	2	1	1	1	1	1
Latency (cyc.)	2	3	3	3	4	6	6	13

It processes one sample per clock cycle and has a latency of two clock cycles.

The proposed radix-3 serial butterfly processes one sample per clock cycle with a latency of four clock cycles.

It requires 9 real adders, 12 real multiplexers, 8 registers, and no real multiplier. Compared to [26], it halves the number of adders from 18 to 9, which is a significant improvement, and also reduces the number of multiplexers by two. This improvement comes at the cost of a slight increase in registers and latency, which is not significant compared to the large reduction in adders. Compared to [27], the proposed approach is more hardware-efficient when processing serial data, as it halves the number of adders and reduces the number of multiplexers and registers by 53% and 33%, respectively. For 2-parallel data, two instances of the proposed butterfly could be used, which would require approximately the same amount of components as [27]. Compared to [28], the proposed radix-3 butterfly uses 5 more real adders and two more registers. However, it saves two real multipliers and two multiplexers. Multipliers are the most hardware-consuming components, being the area of a multiplier similar to the area of a number of adders equal to the data word length. For 16 bits, the two multipliers would require approximately 32 adders, i.e., much more than the adders used in the proposed design.

The proposed radix-4 serial butterfly only requires 4 real adders, 14 real multiplexers, 12 registers, and no real multiplier. It processes one sample per clock cycle and has a latency of six clock cycles.

The proposed radix-5 serial butterfly requires 16 real adders, 45 real multiplexers, and 31 registers, and processes one sample per clock cycle with a latency of 13 clock cycles. Compared to the radix-5 butterfly in [28], the proposed implementation uses 6 more real adders, a similar number of multiplexers, 19 more registers, and takes 7 additional clock cycles to process the inputs. However, it removes the four real multipliers in [28], which are the most hardware-consuming components. For 16-bit data, these multipliers would require around 64 adders, leading to much more hardware cost than in the proposed design. Furthermore, contrary to [28], the proposed approach has the advantage that data are processed in pipeline without feedback loops in the data path. This guarantees that any number of pipeline registers can be added in order to increase the clock frequency.

Compared to the parallel butterflies in Table II, the proposed serial butterflies in Table XI reduce the number of real adders and real multipliers. Regarding the proposed radix-2 and radix-4 butterflies, the number of real adders in the proposed implementations has been reduced by a factor r with respect to the parallel butterflies. This corresponds to a reduction of 50% and 75% for radix-2 and radix-4 butterflies, respectively. Regarding the proposed radix-3 and radix-5 serial butterflies, the multipliers have been reduced by a factor of r. Moreover, these multipliers have been replaced by shift-and-add operations, reducing even more the amount of hardware components. Even when counting the adders in the shift-and-add implementation, the proposed radix-3 and radix-5 serial butterflies reduce the number of real adders by 25% and 53% with respect to the parallel butterflies, respectively. However, some additional logic gates, multiplexers, and registers are included in the proposed serial implementations with respect to the parallel butterflies. In order to take them into account in the comparison, the next section provides experimental results on FPGA and ASIC.

TABLE XII
SQNR OF THE PROPOSED ARCHITECTURES DEPENDING ON THE WORD LENGTH

	SQNR (dB)			
WL	Radix-2	Radix-3	Radix-4	Radix-5
10	53.9	48.7	49.6	43.1
11	60.4	54.6	55.6	49.1
12	66.3	60.6	61.8	55.0
13	72.2	66.5	67.8	60.6
14	78.3	71.7	73.7	65.2
15	84.2	75.3	79.7	67.7
16	90.3	77.4	85.8	68.9
17	96.3	78.0	91.8	69.1
18	102.3	78.0	97.8	69.1

V. EXPERIMENTAL RESULTS

The proposed serial butterflies have been implemented on a Virtex Ultraescale+ HBM XCVU37P-FSVH2892-2L-E. They have been designed with parameterizable word length (WL). The quantization noise in the butterflies has been studied and characterized in Table XII. This table shows the signal-to-quantization-noise ratio (SQNR) [32] in dB as a function of the word length of each real and imaginary part of the data, and assuming that the word length is the same along the circuit. The SQNR is calculated as

$$\text{SQNR (dB)} = 10 \cdot \log_{10}\left(\frac{E\{|X_{ID}|^2\}}{E\{|X_Q - X_{ID}|^2\}}\right), \qquad (8)$$

where $E\{\cdot\}$ represents the expected value, X_{ID} is the output of the ideal FFT without quantization and X_Q is the output of the quantized FFT obtained by the proposed hardware butterflies. For each word length, the experiment considers 1000 trials and uniform distribution of the input data in the full dynamic range defined by these bits.

The experimental results in Table XII show that the SQNR grows at a rate of 6 dB per bit for radix-2 and radix-4 butterflies. For radix-3 and radix-5, the 6 dB increase occurs for small word lengths. However, from $WL = 14$, the quantization noise of the coefficients in the shift-and-add multipliers starts to be significant and it becomes dominant after a word length of 16 bits, where there is a small or no increase in SQNR. Based on this analysis, a word length of 16 bits has been considered for the implementation of all the proposed serial butterflies. Note that $WL = 16$ means 16 bits for the real part of the data and 16 bits for the imaginary part.

Table XIII shows the post-implementation results of the proposed serial butterflies (Prop.) and the parallel butterflies (Par.) in the Virtex Ultrascale+ FPGA. The parallel butterflies have been designed as the direct implementation of their flow graphs shown in Section II. For a fair comparison, both serial and parallel architectures are compared under the same conditions: Every multiplier is implemented with shift-and-add operations, inputs, and outputs are registered, and pipeline registers have been added in order to achieve higher clock frequency. This entails higher latency in terms of clock cycles

TABLE XIII
POST-IMPLEMENTATION RESULTS OF THE PARALLEL BUTTERFLIES (PAR.) AND THE PROPOSED SERIAL BUTTERFLIES (PROP.) ON A VIRTEX XCVU37P-FSVH2892-2L-E

	Radix-2		Radix-3		Radix-4		Radix-5	
Parameter	Par.	Prop.	Par.	Prop.	Par.	Prop.	Par.	Prop.
CLB LUTs	64	80	377	322	264	198	1071	1074
CLB REGs	130	133	614	426	395	330	1637	1068
CARRY8	12	6	56	28	48	12	169	56
CLB	18	21	79	67	62	42	222	197
f_{CLK} (MHz)	650	650	650	650	650	650	650	650
SQNR (dB)	90.3	90.3	77.3	77.3	85.8	85.8	68.8	68.8
Latency (cyc.)	2	4	6	12	3	10	8	23
Power (mW)*	19	19	69	61	63	39	186	150

*: Power consumption measured at 650 MHz

TABLE XIV
POST-SYNTHESYS ASIC RESULTS OF THE PARALLEL BUTTERFLIES (PAR.) AND THE PROPOSED SERIAL BUTTERFLIES (PROP.) USING TSMC 40 NM TECHNOLOGY

	Radix-2		Radix-3		Radix-4		Radix-5	
Parameter	Par.	Prop.	Par.	Prop.	Par.	Prop.	Par.	Prop.
Technology (nm)	40	40	40	40	40	40	40	40
Voltage (V)	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1
Combinational Cells	134	196	1213	1132	698	559	3791	3072
Sequential Cells	130	133	684	462	395	332	1727	1180
Area (μm^2)	790	752	4124	2926	2731	1898	11236	7568
f_{CLK} (MHz)	800	800	800	800	800	800	800	800
SQNR (dB)	90.3	90.3	77.3	77.3	85.8	85.8	68.8	68.8
Latency (cyc.)	2	4	6	12	3	10	8	23
Power (mW)*	1.10	1.11	5.50	3.94	3.64	2.82	14.48	9.93

*: Power consumption measured at 800 MHz

than the values reported in Table XI. The figures of merit included in Table XIII are LUTs, registers, CARRY8s, CLBs, clock frequency, SQNR, latency, and power consumption. The power consumption results consider a clock frequency of 650 MHz.

In Table XIII, it can be observed that the proposed radix-2 serial butterfly uses a larger number of LUTs, a similar number of registers, 3 more CLBs, and half the number of CARRY8s. Compared to the parallel radix-2 butterfly, the proposed one has higher latency than the parallel one and similar power consumption. Considering all these figures of merit, both architectures can be considered similar in terms of hardware resources and performance. However, it is worth noting that the proposed radix-2 butterfly processes data in series, whereas the parallel butterflies process two parallel branches. Therefore, these architectures will be preferable in different scenarios, depending on how data arrive at the butterfly.

Regarding radix-3 butterflies, the proposed serial implementation saves 55 LUTs, 188 registers, and 12 CLBs, and halves the number of CARRY8 compared to the radix-3 parallel butterfly. Likewise, it reduces power consumption by 11%. These improvements in area and power consumption come at the cost of an increase in latency. This increase in latency is an expected result, as the serial butterfly has only one data path to calculate the same operations that a parallel butterfly calculates in parallel, i.e., in the parallel butterflies the operations are distributed among the parallel paths, whereas in the serial butterfly, these operations are distributed in time.

Regarding radix-4 butterflies, the proposed serial butterfly saves 66 LUTs, 65 registers, and 20 CLBs compared to the radix-4 parallel butterfly, which corresponds to savings of 25%, 16%, and 32%, respectively. The reduction of real adders by a factor $r = 4$ causes a reduction of the number of CARRY8 by the same factor. The latency of the proposed approach increases with respect to the parallel radix-4 butterfly and its power consumption is reduced by 38%.

The proposed radix-5 butterfly has a similar number of LUTs compared to the radix-5 parallel butterfly. By contrast, the registers and CLBs are reduced by 27% and 11%, respectively. This reduction in registers is caused by the pipeline registers that are needed in the parallel butterfly so that it can reach a frequency of 650 MHz. i.e., in the proposed serial butterfly a lower amount of registers is needed to reach a frequency of 650 MHz. The proposed radix-5 serial butterfly reduces significantly the number of real adders regarding Table XI, which results in a reduction of CARRY8 by 67%. As expected, the latency of the proposed radix-5 serial butterfly increases and the power consumption decreases with respect to the radix-5 parallel butterfly, leading to savings of 20% in power consumption.

Finally, the proposed serial implementations have the same SQNR as the parallel ones, due to the fact that all the mathematical calculations are the same, including the shift-and-add calculation of the multiplications.

In order to deeply explore the capabilities of the proposed serial designs, ASIC results have been extracted. Table XIV shows the post-synthesis ASIC results for the proposed serial butterflies (Prop.) and the parallel butterflies (Par.) with the same conditions as in Table XIII. The figures of merit included in Table XIV are technology, operating voltage, combinational cells, sequential cells, cell area, SQNR, latency, and power consumption. The power consumption results consider a clock frequency of 800 MHz. The technology used is TSMC of 40 nm. The operational voltage is 1.1 V. The values of SQNR and latency are the same as the ones reported in Table XIII, due to the fact that the logic circuit remains equal in both FPGA and ASIC implementations.

In Table XIV, it can be observed that the proposed radix-2 serial butterfly uses more combinational cells and a similar number of sequential cells. However, the cell area of the proposed radix-2 serial butterfly is slightly smaller. Both implementations have similar power consumption. The experimental results for both radix-2 ASIC implementations are in line with the FPGA results. Regarding the power consumption reported in the radix-2 butterflies FPGA implementations, the radix-2 serial and parallel ASIC implementations reduce the power by 94%.

Regarding radix-3 butterflies, the proposed serial ASIC implementation saves 81 combinational cells and 222 sequential cells, which means a 32% reduction of sequential cells. The area and power consumption are reduced by 29% and 28%, respectively. Regarding the power consumption reported in the radix-3 butterflies FPGA implementations, the radix-3 serial and parallel ASIC implementations reduce the power by 92% and 94%, respectively.

Regarding radix-4 butterflies, the proposed serial ASIC implementation saves 63 combinational cells and 63 sequential cells. The cell area and power consumption are reduced by 31% and 23%, respectively. Regarding the power consumption reported in the radix-4 butterflies FPGA implementations, the radix-4 serial and parallel ASIC implementations reduce the power by 94% and 93%, respectively.

Finally, the proposed radix-5 serial ASIC implementation saves 719 combinational cells and 547 sequential cells, which means a reduction of 19% and 32%, respectively. The total cell area and power consumption are reduced by 33% and 31%, respectively. Regarding the power consumption reported in the radix-4 butterflies FPGA implementations, the radix-4 serial and parallel ASIC implementations reduce the power by 92% and 93%, respectively.

As a result, with the exception of the proposed radix-2 serial butterfly, it can be observed that the proposed serial butterflies reduce area and power by around 30% in the ASIC implementations with respect to the parallel ASIC implementations. The ASIC results reported are in line with the FPGA results, supporting the improvement of the proposed serial designs.

VI. Conclusion

This work has presented new serial butterflies for NP2 FFTs in communication systems for 5G and beyond. Contrary to butterflies in SDF architectures, the serial butterflies proposed in this paper have only one input and one output, which improves their utilization when data are processed in series. Furthermore, the proposed designs distribute efficiently the operations along a pipeline circuit, which reduces the number of hardware components. For radix-2 and radix-4, the proposed serial butterflies achieve the minimum number of real adders and real multipliers. For radix-3 and radix-5, they achieve the minimum number of real multipliers. Additionally, the multipliers have been implemented using shift-and-add operations, which provides further optimization of the circuits.

The proposed circuits have been implemented on an FPGA and ASIC. Their SQNR has been analyzed as a function of the word length. Experimental results show that the proposed serial butterflies achieve a high clock frequency and reduce the area and power consumption with respect to parallel butterflies at the cost of an increase in latency.

The proposed butterflies are suitable for future non-power-of-two serial SC FFT architectures, where the processing elements operate on data that arrive in series in consecutive clock cycles.

Acknowledgment

The authors would like to thank Prof. Martin Kumm for providing the adder graphs of the RMCM multipliers used in the proposed designs.

References

[1] A. Oppenheim and R. Schafer, *Discrete-Time Signal Processing*. Upper Saddle River, NJ, USA: Prentice-Hall, 1989.

[2] *Physical channels and modulation*, document TS 38.211, V16.3.0, 3GPP, Sep. 2020. [Online]. Available: https://www.3gpp.org/DynaReport/38-series.htm

[3] 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, Apr. 1965.

[4] C. M. Rader, "Discrete Fourier transforms when the number of data samples is prime," *Proc. IEEE*, vol. 56, no. 6, pp. 1107–1108, Jun. 1968.

[5] S. Winograd, "On computing the discrete Fourier transform," *Math. Comput.*, vol. 32, no. 141, pp. 175–199, Jan. 1978.

[6] C. Burrus, "Index mappings for multidimensional formulation of the DFT and convolution," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-25, no. 3, pp. 239–242, Jun. 1977.

[7] S. K. Mitra, *Discrete Signal Processing: A Computer-Based Approach*, 4th ed. New York, NY, USA: McGraw-Hill, 2011.

[8] H. J. Nussbaumer, *The Fast Fourier Transform and Convolution Algorithms*, 2nd ed. Berlin, Germany: Springer, 1982.

[9] S.-C. Hsu, S.-J. Huang, S.-G. Chen, S.-C. Lin, and M. Garrido, "A 128-point multi-path SC FFT architecture," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Oct. 2020, pp. 1–5.

[10] M. Garrido, S.-J. Huang, S.-G. Chen, and O. Gustafsson, "The serial commutator FFT," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 63, no. 10, pp. 974–978, Oct. 2016.

[11] A. Chinnapalanichamy and K. K. Parhi, "Serial and interleaved architectures for computing real FFT," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Apr. 2015, pp. 1066–1070.

[12] M. Garrido, F. Qureshi, J. Takala, and O. Gustafsson, "Hardware architectures for the fast Fourier transform," in *Handbook of Signal Processing Systems*, S. S. Bhattacharyya, E. F. Deprettere, R. Leupers, and J. Takala, Eds., 3rd ed. Berlin, Germany: Springer, 2019.

[13] M. Garrido, R. Andersson, F. Qureshi, and O. Gustafsson, "Multiplier-less unity-gain SDF FFTs," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 24, no. 9, pp. 3003–3007, Sep. 2016.

[14] A. Cortés, I. Vélez, and J. F. Sevillano, "Radix r^k FFTs: Matricial representation and SDC/SDF pipeline implementation," *IEEE Trans. Signal Process.*, vol. 57, no. 7, pp. 2824–2839, Jul. 2009.

[15] L. Yang, K. Zhang, H. Liu, J. Huang, and S. Huang, "An efficient locally pipelined FFT processor," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 53, no. 7, pp. 585–589, Jul. 2006.

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

[17] M. Garrido, "Evolution of the performance of pipelined FFT architectures through the years," in *Proc. 35th Conf. Design Circuits Integr. Syst. (DCIS)*, Nov. 2020, pp. 1–6.

[18] I. Cho, T. Patyk, D. Guevorkian, J. Takala, and S. Bhattacharyya, "Pipelined FFT for wireless communications supporting 128–2048/1536-point transforms," in *Proc. IEEE Global Conf. Signal Inf. Process.*, Dec. 2013, pp. 1242–1245.

[19] J. Kim, J. Lee, and K. Cho, "A mixed-radix pipeline FFT processor with trivial multiplications for LTE uplink," in *Proc. IEEE Int. Symp. Consum. Electron. (ISCE)*, Sep. 2016, pp. 57–58.

[20] J. Löfgren, L. Liu, O. Edfors, and P. Nilsson, "Improved matching-pursuit implementation for LTE channel estimation," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 61, no. 1, pp. 226–237, Jan. 2014.

[21] 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.

[22] 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.

[23] X.-Y. Shih, Y.-Q. Liu, and H.-R. Chou, "48-mode reconfigurable design of SDF FFT hardware architecture using radix-3^2 and radix-2^3 design approaches," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 64, no. 6, pp. 1456–1467, Jun. 2017.

[24] C.-H. Yang, T.-H. Yu, and D. Markovic, "Power and area minimization of reconfigurable FFT processors: A 3GPP-LTE example," *IEEE J. Solid-State Circuits*, vol. 47, no. 3, pp. 757–768, Mar. 2012.

[25] X. Yang, W. Du, Z. Yang, R. Lv, and X. Wang, "Hardware design and implementation of 3780 points FFT based on FPGA in DTTB," in *Proc. 4th Int. Conf. Appl. Inf. Commun. Technol.*, Oct. 2010, pp. 1–4.

[26] C. Yu and M.-H. Yen, "Area-efficient 128- to 2048/1536-point pipeline FFT processor for LTE and mobile Wimax systems," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 23, no. 9, pp. 1793–1800, Sep. 2015.

[27] W.-L. Tsai, S.-G. Chen, and S.-J. Huang, "Reconfigurable radix-$2^k \times 3$ feedforward FFT architectures," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2019, pp. 1–5.
[28] G.-T. Deng, "A radix-2^k multi-path serial conmutator and $2^{\alpha} \cdot 3^{\beta} \cdot 5^{\gamma}$-point serial conmutator FFT processor," M.S. thesis, Inst. Electron., Nat. Yang Ming Chiao Tung Univ., Hsinchu, Taiwan, Sep. 2021.
[29] V. M. Bautista, "Design and implementation of FFT architectures with non-power-of-two sizes for 5G," Bachelor's thesis, Dept. Electron. Eng., Univ. Politécnica de Madrid, Madrid, Spain, Jul. 2021.
[30] 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.
[31] K. Möller, M. Kumm, M. Kleinlein, and P. Zipf, "Reconfigurable constant multiplication for FPGAs," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 36, no. 6, pp. 927–937, Jun. 2017.
[32] D. Guinart, "Deterministic analysis of the accuracy in FFT hardware architectures," M.S. thesis, Dept. Electron. Eng., Linköping Univ., Linköping, Sweden, Jun. 2012.

Víctor Manuel Bautista was born in Madrid, Spain, in 1998. He received the B.E. degree in telecommunication technologies and services engineering and the master's degree in electronic systems engineering (MUISE) from Universidad Politécnica de Madrid (UPM), Madrid, in July 2021 and July 2022, respectively, where he is currently pursuing the Ph.D. degree with the Department of Electronic Engineering. His research interests include optimized hardware design for communication systems, focusing on the design of fast Fourier transform (FFT) hardware architectures for non-power-of-two sizes.

Mario Garrido (Senior Member, IEEE) received the M.Sc. and Ph.D. degrees in electrical engineering from Universidad Politécnica de Madrid (UPM), 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. In 2022, he appeared in the "World's Top 2% Scientists List" elaborated by Stanford University. 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.

Marisa López-Vallejo (Senior Member, IEEE) received the M.S. and Ph.D. degrees from Universidad Politécnica de Madrid (UPM), Madrid, Spain, in 1993 and 1999, respectively. From 2015 to 2016, she was a Visiting Professor with the Microsystems Technology Laboratory, Massachusetts Institute of Technology (MIT), Cambridge, MA, USA. She was with Bell Laboratories, Lucent Technologies, Murray Hill, NJ, USA, as a Technical Staff Member. Since 2016, she has been a Full Professor with the Department of Electronic Engineering, UPM. Her research interests include low-power, radiation, and PVT-aware design, computer-aided diagnostic methods and tools, and application-specific high-performance programmable architectures. In last decade she has focused her research on the reliability of CMOS circuits and memristive memories as well as on new architectures to support reliable design beyond 20-nm. She has been a coordinator of a set of national and international projects in these areas.