A Method to Obtain Non-Power-of-Two FFT Flow Graphs Based on a New Prime Factor Algorithm

Víctor Manuel Bautista and Mario Garrido, *Senior Member, IEEE*

***Abstract*—This paper presents a novel method to obtain non-power-of-two (NP2) fast Fourier transform (FFT) flow graphs based on a new prime factor algorithm (PFA). The FFT flow graph is crucial for designing FFT architectures but previous works only provide systematic approaches to build flow graphs for power-of-two sizes (P2). Thus, the derivation of NP2 flow graphs is an important step towards the design of efficient NP2 FFT architectures.**

The proposed approach consists of two independent parts. On the one hand, it obtains all the possible index mappings that lead to a flow graph with no rotations between butterflies. On the other hand, it determines the permutations between butterflies in the flow graph. By combining these two parts, the order of the inputs and outputs is derived. As a result, the entire flow graph is obtained systematically. Additionally, the proposed approach generates all the possible flow graphs for a given factorization of the FFT size.

The reduction in operations for NP2 FFTs using the proposed approach leads to a significant reduction in area and power consumption concerning P2 FFTs with similar sizes after implementing the proposed flow graphs directly in hardware. Particularly, there is a significant improvement between the proposed 30-point and 60-point FFT and previous efficient P2 FFTs. This remarkable fact sets NP2 at the forefront of FFT research after being in second place behind P2 FFTs for decades.

***Index Terms*—Fast Fourier transform (FFT), non-power-of-two (NP2), index mapping, prime factor algorithm (PFA), butterfly.**

I. Introduction

THE discrete Fourier transform (DFT) serves as a crucial element in digital communications. This mathematical formula converts the signal representation from the time domain into the frequency domain. The fast Fourier transform (FFT) algorithm, initially proposed by Cooley and Tukey [1] divides the original DFT into smaller DFTs and twiddle factors. The butterflies calculate the small DFTs and the twiddle factors carry out rotations in the complex plane. The FFT algorithm significantly reduces the computational complexity from $\mathcal{O}(N^2)$ in the DFT to a more efficient $\mathcal{O}(N \log N)$.

In 5G communication systems, the FFT sizes are a product of powers of two, three, and five [2], which involves non-power-of-two (NP2) FFT sizes. Whereas the FFTs whose sizes are a power of two (P2) have been deeply explored in the literature [3]–[5], leading to efficient FFT architectures, NP2 FFTs have not reached the same development. The reason is that specific FFT algorithms for NP2 sizes are more complex than the Cooley-Tukey algorithm. The difficulty of using these algorithms has impeded its application to design efficient NP2 FFT architectures.

FFT algorithms for NP2 sizes were mostly developed during the last decades of the 20th century. Among them, Rader's FFT algorithm [6] converts the FFT into a convolution by a specific index mapping when the size is a prime number. We call index mapping to the expression used to obtain the input and output sequence orders of a certain FFT. Winograd's algorithm [7] continued this topic by designing butterflies that reduce the number of multiplications. These butterflies are still used [8]–[11]. This approach can be combined with the Cooley-Tukey algorithm to reduce the complexity of the FFT [10].

When the size of the FFT consists of a product of coprime numbers, alternative index mappings are used to eliminate the rotations calculated between the butterflies. This approach is called prime factor algorithm (PFA). A wide range of prime factor FFT algorithms have been proposed [12]–[15]. Good's PFA [12] presented the first solution. His approach applied advanced congruence theory to express the input and output indexes in a linear form. This form consists of a sum of residues multiplied by coefficients and expressed as a congruence. Then, Burrus and Eschenbacher's PFA [13] generalized the approach of [12] by providing a parametric solution for the index mapping. This solution is based on changing the rotations that are calculated inside the butterflies. Temperton's PFA [14] details the implementation of the butterflies that lead to any rotation proposed in [13]. Temperton's PFA [14] also proposed a new index mapping and an efficient software implementation to calculate the input and output orders. Both [13], [14] algorithms consider different rotations inside the butterflies, which means that different butterflies for the same size have to be designed. The approach of Lun and Siu [15] considers an index mapping for the realization of the PFA with standard butterflies, solving the issue of [13] and [14]. However, the expression of the input and output indexes requires the calculation of square roots evaluated under modulo. This operation only has a solution under certain circumstances, depending on the factorization of the FFT's size. This limits the amount of valid FFT sizes. Finally, [16] presents the schemes to develop in-place and in-order prime factor algorithms, and provides a survey of previous index mapping approaches.

Since these approaches were presented, for more than two decades there have not been remarkable advancements related to NP2 FFT algorithms. As a consequence of this, the advancement in NP2 FFT architectures has also been very limited. Related to FFT architectures, one research line that has

V. M. Bautista and M. Garrido are with the Department of Electronic Engineering, ETSI de Telecomunicación, Universidad Politécnica de Madrid (UPM), 28040 Madrid, Spain, e-mails: victor.bautista@upm.es, mario.garrido@upm.es

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.

been followed has been to add an NP2 stage to a P2 FFT [17]–[20]. These works consider sizes of the form $2^k \cdot 3$. This is a specific case of NP2 FFT that does not require solving the big challenge of using NP2 algorithms. Other works consider the Cooley-Tukey algorithm for NP2 FFTs [21], [22] similarly as it is used for P2 algorithms. The main advantage of the PFA with respect to Cooley-Tukey for NP2 is that it removes rotations between butterflies. As the Cooley-Tukey algorithm does not remove these rotations, NP2 architectures based on the Cooley-Tukey algorithm include rotations both in the butterflies and between butterflies, which causes an overhead in the area of the architectures. Another approach that has been considered in the literature is the calculation of variable-length FFTs for a large number of P2 and NP2 FFT sizes [23]–[25]. Finally, only a few FFT architectures in the literature are based on the PFA [26]–[29]. The reduced number of works in this area and the lack of solid knowledge about NP2 FFTs make it crucial to deepen into this research field and obtain new fundamental knowledge that helps to move forward in a research area that has been stuck for decades.

One of the key advancements that have permitted the development of P2 FFTs has been the deep understanding of the flow graphs of P2 FFT algorithms [5]. This has allowed us to derive models and rules for the architectures, making it possible to design numerous different types of FFT architectures for P2 sizes [4], [30]. By contrast, a systematic approach to derive the flow graph of an NP2 FFT has not been proposed in the literature so far, which hinders the development of NP2 FFT architectures.

This paper arises from the need to obtain the flow graph of NP2 FFT algorithms to make the design of efficient FFT hardware architectures feasible. With this purpose, this paper presents a method to derive NP2 flow graphs based on a new index mapping for the prime factor algorithm. The objective is to obtain all the possible flow graphs from a given factorization of the FFT size in coprime factors. In this work, the derivation of the flow graphs is divided into two independent problems. The first one is to obtain the formula that provides the coefficients of the index mapping of the algorithm. The requirement is that it must work with any factorization of coprime numbers and the butterflies must be standard, i.e., each radix-r butterfly in the flow graph must calculate an r-point FFT. The second problem is to formally describe a procedure to obtain the connections between butterflies in the flow graph. The connections are characterized by the digit representation of the data index along the flow graph. As a result, the proposed approach provides a systematic method to obtain NP2 flow graphs and can obtain all the flow graphs for a given factorization of the FFT size into coprime numbers. This paper also extends the procedure to the case when the size of the FFT includes factors that are not coprime. Thus, the combination of the proposed methodology with the Cooley-Tukey algorithm can decompose any FFT size efficiently.

By comparing the number of operations of the proposed flow graphs with previous ones, it is observed that the use of the prime factor algorithm reduces significantly the number of operations with respect to Cooley-Tukey for NP2 sizes. In fact, it has been observed that many NP2 FFT sizes close to a P2 size require considerably fewer operations than the corresponding P2 size. This effect is supported by the hardware implementation of fully parallel (FP) FFT architectures [31]–[33], which are the direct implementation of the flow graphs. The FPGA implementation of the proposed architectures demonstrates significant savings of area and power consumption in cases such as 30-point and 60-point FFTs compared to efficient 32-point and 64-point FFT architectures, respectively. This is a remarkable result supporting that NP2 FFTs are not only feasible but also that they can beat P2 ones.

This paper is organized as follows: In Section II, the state-of-the-art is reviewed. In Section III, the problem that is solved in this paper is stated. In Section IV, the methodology to obtain the flow graphs for NP2 FFTs is presented. In Section V, the proposed work is compared with previous ones. In Section VI, experimental results of the proposed work are provided and compared to previous works. Finally, in Section VII, the main conclusions of this work are summarized.

II. Background

A. The FFT Algorithm

An N-point discrete Fourier transform of a discrete complex signal $x[n]$ is calculated as

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

where $X[k]$ represents the output at frequency k. The terms $W_N^{nk} = e^{-j\frac{2\pi}{N}nk}$ are called twiddle factors and calculate rotations 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 elements that calculate these small DFTs are called butterflies. A radix-r butterfly calculates an r-point DFT.

Let us consider a DFT whose size is split into M factors as $N = N_1 \cdot N_2 \cdot \ldots \cdot N_M$. Then, the input index, n, and the output index, k, from (1) can be expressed by the index mapping [34]

$$n \equiv \alpha_1 n_1 + \alpha_2 n_2 + ... + \alpha_M n_M \pmod N, \quad (2a)$$
$$k \equiv \beta_1 k_1 + \beta_2 k_2 + ... + \beta_M k_M \pmod N, \quad (2b)$$

where α_s and β_s for $s = 1, \ldots, M$ are scalar values and $n_s, k_s = 0, 1, \ldots, N_s - 1$ are the residues of N_s. The values of n and k are evaluated modulo N with a congruence operation. The selection of the coefficients α_s and β_s leads to different inputs, outputs and rotations. In case of the Cooley-Tukey algorithm [1], it selects the coefficients as

$$\alpha_s = \frac{N}{\prod_{j=1}^{s} N_j}, \quad (3a)$$

$$\beta_s = \frac{N}{\prod_{j=1}^{M-s+1} N_{M-j+1}}. \quad (3b)$$

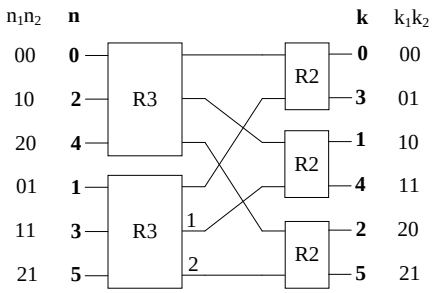

Fig. 1. Example of 6-point FFT using the Cooley-Tukey algorithm.

For example, if $N = 6$, with $N_1 = 3$ and $N_2 = 2$, the index mapping according to the Cooley-Tukey algorithm is

$$n \equiv 2 \cdot n_1 + n_2 \pmod 6, \tag{4a}$$
$$k \equiv k_1 + 3 \cdot k_2 \pmod 6. \tag{4b}$$

Fig. 1 shows the flow graph of the 6-point FFT using the Cooley-Tukey algorithm. We can observe 2 radix-3 butterflies and 3 radix-2 butterflies denoted by R3 and R2, respectively. Between the butterflies, we can see the numbers 1 and 2. They represent rotations by W_6^1 and by W_6^2, respectively. Inputs are represented by the digit-representation (n_1, n_2), which is multiplied by the coefficients vector $(2, 1)$ to obtain the decimal values n, according to (4a). The digit n_1 corresponds to the residues of the first stage and the digit n_2 corresponds to the residues of the second stage. The digit-representation of the inputs of the last stage, (k_1, k_2), multiplied by the output coefficients $(1, 3)$ leads to k, according to (4b). The digit k_1 corresponds to the residues of the first stage and the digit k_2 corresponds to the residues of the second stage.

B. Prime Factor Algorithm

The prime factor algorithm [12] eliminates the twiddle factors that appear in the Cooley-Tukey algorithm. The condition is that N must be divided into a product of M numbers N_i, $i = 1, ..., M$ that are coprime, i.e., $N = N_1 \cdot N_2 \cdot \ldots \cdot N_M$. Two numbers are coprime if they do not share common factors. This algorithm uses the index mapping model in (2a) and (2b). Several PFA solutions to this index mapping have been proposed in the literature. They differ in the coefficients α_s and β_s. In [12], the Ruritarian correspondence for n and the Sino correspondence for k were proposed, which corresponds to

$$n \equiv \frac{N}{N_1} n_1 + ... + \frac{N}{N_M} n_M \pmod N, \tag{5a}$$
$$k \equiv \frac{N}{N_1} \left(\frac{N}{N_1}\right)^{-1}_{N_1} k_1 + ... + \frac{N}{N_M} \left(\frac{N}{N_M}\right)^{-1}_{N_M} k_M \pmod N. \tag{5b}$$

The operator $(a)_b^{-1}$ represents the inverse modulo of a. In congruence theory, it can be calculated by clearing x in the congruence $ax \equiv 1 \pmod b$ [35]. The Sino correspondence is derived using the Chinese Remainder Theorem (CRT) [36]. By using this index mapping, the twiddle factors between butterflies are removed from the flow graph.

Another index mapping for the PFA is proposed in [13]. This work generalizes the solution of [12] as

$$n \equiv \left(a_1 \frac{N}{N_1} + ... + a_M \frac{N}{N_M}\right) \cdot k \pmod N, \tag{6a}$$
$$k \equiv \frac{a_1 N}{N_1} \left(\frac{N}{N_1}\right)^{-1}_{N_1} k_1 + . + \frac{a_M N}{N_M} \left(\frac{N}{N_M}\right)^{-1}_{N_M} k_M \pmod N. \tag{6b}$$

This work uses a generalization of the CRT and the parameters a_s to derive k. The parameters a_s are chosen so that a_s and N_s are coprime, leading to different possible solutions. Note that if $a_s = 1$ we obtained the expression of (5b). The input index mapping of n is obtained directly by multiplying the mapping of k by an escalar value. (6a) is dependant of the digits k_s, so the result of n still has to be unscrambled according to n_s, which adds an additional step. Even though this index mapping proposes several solutions that eliminate the twiddle factors, some of these solutions require changing the rotations inside some butterflies. This has two implications. First, butterflies of the same size are not standard, i.e., they do not calculate a DFT of the size of the butterfly. Second, the butterflies are not homogeneous, and different types of butterflies are needed in this approach.

The work presented in [14] assumed all the possible rotations that can be performed in the butterflies. This algorithm describes the index mapping as

$$n \equiv \frac{N}{N_1} \left(\frac{N}{N_1}\right)^{-1}_{N_1} n_1 + ... + \frac{N}{N_M} \left(\frac{N}{N_M}\right)^{-1}_{N_M} n_M \pmod N, \tag{7a}$$
$$k \equiv \frac{N}{N_1} \left(\frac{N}{N_1}\right)^{-1}_{N_1} k_1 + ... + \frac{N}{N_M} \left(\frac{N}{N_M}\right)^{-1}_{N_M} k_M \pmod N. \tag{7b}$$

Note that input and output coefficients are the same, which facilitates the extraction of α_s and β_s. Still, the algorithm leads to the same problem as [13]: The butterflies are not homogeneous.

The approach presented in [15] considered using the standard structure of the butterflies while using the same input and output coefficients. The index mapping is described as

$$n \equiv \sum_{s=1}^{M} \left(\sqrt{\left(\frac{N}{N_s}\right)^{-1}}_{N_s}\right)_{N_s} \frac{N}{N_s} n_s \pmod N, \tag{8a}$$
$$k \equiv \sum_{s=1}^{M} \left(\sqrt{\left(\frac{N}{N_s}\right)^{-1}}_{N_s}\right)_{N_s} \frac{N}{N_s} k_s \pmod N. \tag{8b}$$

This avoids the problem of having different butterfly designs for different factorizations. However, to use the index mapping proposed in [15], the square root modulo N_s must exist, which is not always the case. Thus, this approach is only valid for a subset of FFT sizes.

III. Problem Statement

Fig. 2 shows the structure of an FFT flow graph. The information about input order, output order, and connections between butterflies has been omitted. Different permutations

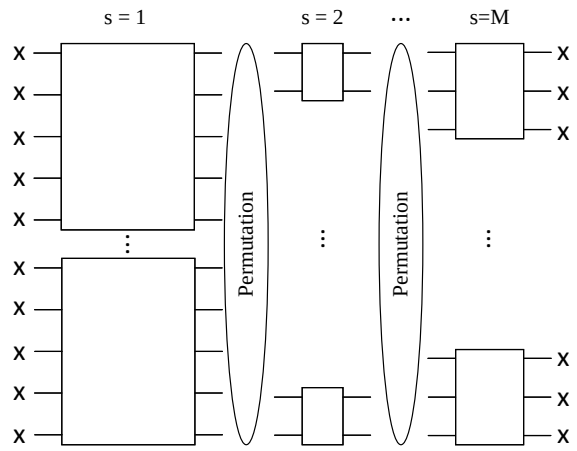

Fig. 2. Problem statement for an FFT with three stages.

lead to different connections between butterflies. Different coefficients of the index mapping lead to different rotations between stages and to a different input and output pattern, which is represented with 'X' in Fig. 2. The aim of this paper is to obtain all the possible flow graphs for a given factorization of N. This challenge is divided into two subproblems. The first one is to obtain the coefficients for the index mapping formulas of (2a) and (2b), so that twiddle factors are removed from the flow graph and all the butterflies are standard and calculate an FFT of their size. The second subproblem aims to obtain the permutations that connect the butterflies in different stages. This work also quantifies the proposed number of PFA solutions and the number of possible permutations for a certain factorization.

IV. Proposed Approach

This section presents a new PFA that overcomes the limitations of previous algorithms. Specifically, the proposed approach obtains the input and output coefficients α_s and β_s with an index mapping that uses standard butterflies, it is valid for any N that is the product of coprime numbers, and it can obtain all the existing flow graphs.

A. Obtaining the Input and Output Coefficients

Let us consider an N-point DFT, split into two factors, N_1 and N_2, as $N = N_1 \cdot N_2$. Using the general form of the index mapping in (2), (1) is reformulated as

$$X[k] = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} x[n] \cdot W_N^{nk}, \tag{9}$$

with

$$n \equiv \alpha_1 n_1 + \alpha_2 n_2 \pmod N, \tag{10a}$$
$$k \equiv \beta_1 k_1 + \beta_2 k_2 \pmod N, \tag{10b}$$

where $n, k = 0, 1, \ldots, N-1$ and n_s and k_s are the residues of N_s, which take the values $n_s, k_s = 0, 1, \ldots, N_s - 1$. The expression W_N^{nk} represents the rotations and can be expanded as

$$W_N^{nk} = W_N^{\alpha_1\beta_1 n_1 k_1} \cdot W_N^{\alpha_1\beta_2 n_1 k_2} \cdot W_N^{\alpha_2\beta_1 n_2 k_1} \cdot W_N^{\alpha_2\beta_2 n_2 k_2}. \tag{11}$$

The term $W_N^{\alpha_1\beta_1 n_1 k_1}$ is associated with the rotations calculated in the butterflies of the first stage. The terms $W_N^{\alpha_1\beta_2 n_1 k_2}$ and $W_N^{\alpha_2\beta_1 n_2 k_1}$ are associated to the rotations calculated between butterflies. The term $W_N^{\alpha_2\beta_2 n_2 k_2}$ is associated with the rotations calculated in the butterflies of the second stage. Following the conventional prime factor algorithm [12], [34], the conditions to remove the twiddle factors from the flow graph are

$$W_N^{\alpha_1\beta_2 n_1 k_2} = 1, \tag{12a}$$
$$W_N^{\alpha_2\beta_1 n_2 k_1} = 1, \tag{12b}$$

so that there are no rotations between the stages. Equations (12a) and (12b) are satisfied if

$$\alpha_1\beta_2 \equiv 0 \pmod N, \tag{13a}$$
$$\alpha_2\beta_1 \equiv 0 \pmod N. \tag{13b}$$

At the same time, the rotations calculated in the butterflies are considered to be the rotations of a standard DFT as a requirement. Following this, we state that

$$W_N^{\alpha_1\beta_1 n_1 k_1} = W_{N_1}^{n_1 k_1}, \tag{14a}$$
$$W_N^{\alpha_2\beta_2 n_2 k_2} = W_{N_2}^{n_2 k_2}, \tag{14b}$$

so that (14a) corresponds to the rotations calculated in a radix-N_1 butterfly and (14b) corresponds to the rotations calculated in a radix-N_2 butterfly. The relation that satisfies (14a) and (14b) is

$$\alpha_1\beta_1 \equiv \frac{N}{N_1} \pmod N, \tag{15a}$$
$$\alpha_2\beta_2 \equiv \frac{N}{N_2} \pmod N. \tag{15b}$$

From the conditions (13a), (13b), (15a), and (15b), we can observe that, once the coefficients α_s are known, the coefficients β_s are directly calculated. Analogously, knowing the coefficients β_s, the coefficients α_s can also be directly calculated. For this procedure, we start obtaining the input coefficients and then we obtain the output coefficients. The requirement is that a unique solution is needed for a certain data distribution. This means that, given a signal flow graph as in Fig. 1, input and output indexes can not be repeated. Also, as there are N input and output indexes that can not be repeated, all the inputs and outputs must be covered. Equation [34] states that, considering the linear form of the index mapping in (2), the sufficient condition for the mapping to be unique is that α_s is coprime to N_s and is proportional to $\frac{N}{N_s}$. Then, the solution is derived by the Chinese Remainder Theorem [36] as follows. Let us consider the notation used in (2a), where N is factorized in M coprime factors. The CRT states that, given a congruence system where the number n is known by the residues modulo N_s as

$$n \equiv n_1 \pmod{N_1}, \tag{16a}$$
$$n \equiv n_2 \pmod{N_2}, \tag{16b}$$
$$\vdots \tag{16c}$$
$$n \equiv n_M \pmod{N_M}, \tag{16d}$$

then n can be uniquely determined modulo $N = N_1 \cdot N_2 \cdot \ldots \cdot N_M$ as

$$n \equiv \sum_{s=1}^{M} \left(\frac{N}{N_s} \right)_{N_s}^{-1} \frac{N}{N_s} n_s \pmod{N}. \tag{17}$$

The inverse modulo has unique solution if $\frac{N}{N_s}$ and N_s are coprime. The generalization of the CRT has been proposed in [13]. The generalization of CRT applied to an input index mapping with two coefficients has the form

$$n \equiv a_1 \left(\frac{N}{N_1} \right)_{N_1}^{-1} \frac{N}{N_1} n_1 + a_2 \left(\frac{N}{N_2} \right)_{N_2}^{-1} \frac{N}{N_2} n_2 \pmod{N}. \tag{18}$$

Equation (18) includes the parameters a_s with respect (17). These parameters can be chosen so that a_s are coprime to N_s. The election of a_s leads to different solutions of the PFA. Each of the solutions gives a unique distribution of the index map. According to the conditions in [34], the output map is also necessary to be a solution of the generalization of the CRT as

$$k \equiv b_1 \left(\frac{N}{N_1} \right)_{N_1}^{-1} \frac{N}{N_1} k_1 + b_2 \left(\frac{N}{N_2} \right)_{N_2}^{-1} \frac{N}{N_2} k_2 \pmod{N}, \tag{19}$$

which involves the parameters b_s, analogous to a_s. From (18), and (19) we can identify the input and output coefficients as

$$\alpha_s \equiv a_s \left(\frac{N}{N_s} \right)_{N_s}^{-1} \frac{N}{N_s} \pmod{N}, \tag{20a}$$

$$\beta_s \equiv b_s \left(\frac{N}{N_s} \right)_{N_s}^{-1} \frac{N}{N_s} \pmod{N}, \tag{20b}$$

for $s = \{1, 2\}$ in this procedure. The problem of finding the coefficients α_s and β_s is then reduced to finding the proper values for a_s and b_s so that the conditions (13a), (13b), (15a), and (15b) are satisfied.

Note that, for any value of a_s, the value of α_s will always be proportional to $\frac{N}{N_s}$. Analogously, for any value of b_s, the value of β_s will always be proportional to $\frac{N}{N_s}$. Therefore, any product $\alpha_i \cdot \beta_j$ with $i \neq j$, is proportional to N despite the value of a_i and b_j. This means that using the CRT to obtain input and output coefficients does automatically satisfy (13a) and (13b). Then, the relation between a_s and b_s, if it exists, is extracted only from conditions (15a) and (15b). However, (15a) and (15b) do not have a direct solution, as α_s and β_s are not coprime with respect to N. The conditions (15a) and (15b) can be further expanded with congruence properties. By substituting (20a) and (20b) in the general case of (15a) and (15b), the expression becomes

$$a_s \cdot b_s \cdot \left(\frac{N}{N_s} \right)_{N_s}^{-2} \left(\frac{N}{N_s} \right)^2 \equiv \frac{N}{N_s} \quad \left(\text{mod } N_s \frac{N}{N_s} \right), \tag{21}$$

where $(x)_m^{-h}$ is an inverse modulo to the power of h evaluated modulo m. The term $\left(\frac{N}{N_s} \right)_{N_s}^2$ can not be directly multiplied with $\left(\frac{N}{N_s} \right)^2$ to become the unity because the first term is evaluated modulo N_s and the second one is evaluated modulo N in the congruence. In order to simplify (21), let us consider the congruence

$$a \cdot c \equiv b \cdot c \pmod{m}. \tag{22}$$

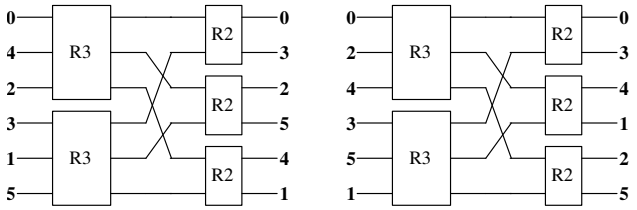

Fig. 3. Two different flow graphs from the proposed PFA for a 6-point FFT. (a) $\{a_1, a_2\} = \{1, 1\}$. (b) $\{a_1, a_2\} = \{2, 1\}$.

If a and b do not share common factors with m but c does, (22) is simplified as

$$a \equiv b \quad \left(\text{mod } \frac{m}{\gcd(m, c)} \right), \tag{23}$$

where $\gcd(\cdot)$ is the greatest common divisor between two numbers. As α_s and β_s are always proportional to $\frac{N}{N_s}$, the factor $\frac{N}{N_s}$ always appears in the left term of (21). From conditions (15), the parameter $\frac{N}{N_s}$ necessarily appears in the right term of the congruence by definition. As $\frac{N}{N_s}$ is necessarily a factor of N, the property of (23) can be applied to (21) as

$$a_s \cdot b_s \cdot \left(\frac{N}{N_s} \right)_{N_s}^{-2} \frac{N}{N_s} \equiv 1 \pmod{N_s}. \tag{24}$$

Note that (24) is now evaluated modulo N_s so that a certain number multiplied by its inverse modulo N_s becomes the unity. The relation between a_s and b_s is given by

$$a_s \cdot b_s \cdot \left(\frac{N}{N_s} \right)_{N_s}^{-1} \equiv 1 \pmod{N_s}, \tag{25}$$

and, by clearing b_s, we obtain

$$b_s \equiv (a_s)_{N_s}^{-1} \frac{N}{N_s} \pmod{N_s}, \tag{26}$$

It is important to highlight that b_s has a unique solution only if a_s is coprime with respect to N_s. This means that values of a_s that share common factors with respect to N_s are not allowed to be chosen, which sets the condition $\gcd(a_s, N_s) = 1$. As a result, the proposed expressions for the input and output coefficients are

$$\alpha_s \equiv a_s \left(\frac{N}{N_s} \right)_{N_s}^{-1} \frac{N}{N_s} \pmod{N}, \quad \gcd(a_s, N_s) = 1, \tag{27a}$$

$$\beta_s \equiv (a_s)_{N_s}^{-1} \left(\frac{N}{N_s} \right)_{N_s}^{-1} \left(\frac{N}{N_s} \right)^2 \pmod{N}. \tag{27b}$$

As an example, Fig. 3(a) and 3(b) represent two different flow graphs of a 6-point FFT. Each flow graph is associated with one of the two possible solutions of the proposed prime factor algorithm for a 6-point FFT. Fig. 3(a) uses $a_1 = 1$ and $a_2 = 1$, so its index mapping is given by

$$n \equiv 4 \cdot n_1 + 3 \cdot n_2 \pmod{6}, \tag{28a}$$

$$k \equiv 2 \cdot k_1 + 3 \cdot k_2 \pmod{6}. \tag{28b}$$

Fig. 3(b) uses $a_1 = 2$ and $a_2 = 1$, so its index mapping is given by

$$n \equiv 2 \cdot n_1 + 3 \cdot n_2 \pmod 6, \tag{29a}$$
$$k \equiv 4 \cdot k_1 + 3 \cdot k_2 \pmod 6. \tag{29b}$$

It can be observed that both figures differ in the input and output orders, though both algorithms implement the same number of operations.

Finally, we can generalize the proposed PFA index mapping as follows. Let us consider an N-point DFT split into M factors as $N = N_1 \cdot N_2 \cdot \ldots \cdot N_M$. By using the general form of the index mapping in (2), (1) is reformulated as

$$X[k] = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} \cdots \sum_{n_M=0}^{N_M-1} x[n] \cdot W_N^{nk}, \tag{30}$$

with

$$n \equiv \sum_{s=1}^{M} \alpha_s n_s \pmod N, \tag{31a}$$
$$k \equiv \sum_{s=1}^{M} \beta_s k_s \pmod N. \tag{31b}$$

Analogously to the case of two factors, the condition to remove the twiddle factors is

$$W_N^{\alpha_i \beta_j n_i k_j} = 1, \tag{32}$$

which must hold for every $i \neq j$. The condition (32) is satisfied if

$$\alpha_i \beta_j \equiv 0 \pmod N, \tag{33}$$

which must hold for every $i \neq j$. Choosing the CRT form of the input and output index mapping makes α_i proportional to $\frac{N}{N_i}$ and also makes β_j proportional to $\frac{N}{N_j}$. Thus, the product will always be proportional to N and condition (32) is automatically satisfied.

At the same time, to have M standard butterflies, the relation that must be satisfied is

$$\alpha_s \beta_s \equiv \frac{N}{N_s} \pmod N, \tag{34}$$

for every $s = 1, 2, \ldots, M$. Note that (34) only concerns coefficients of the same stage s. Thus, exactly the same procedure as it was detailed for $s = 1, 2$ can be applied to obtain the coefficients when $s = 1, 2, \ldots, M$. Therefore, the proposed index mapping for a general factorization of N is obtained by substituting expressions (27a) and (27b), considering $s = 1, 2, \ldots, M$, into (31a) and (31b) as

$$n \equiv \sum_{s=1}^{M} a_s \left(\frac{N}{N_s}\right)_{N_s}^{-1} \frac{N}{N_s} n_s \pmod N, \quad \gcd(a_s, N_s) = 1, \tag{35a}$$

$$k \equiv \sum_{s=1}^{M} (a_s)_{N_s}^{-1} \left(\frac{N}{N_s}\right)_{N_s}^{-1} \left(\frac{N}{N_s}\right)^2 k_s \pmod N. \tag{35b}$$

As a_s and N_s must not share common factors, the range of a_s that can be chosen is limited. For each stage s, the number of coprime numbers to N_s is given by the Euler's Totient function $\varphi(N_s)$ [35]. This function returns the number of valid a_s that can be chosen for a certain stage s. For example, $\varphi(6) = 2$ as only the numbers 1 and 5 do not share common factors with 6, which makes them possible values of a_s if $N_s = 6$. Furthermore, as 6 consists of a product of coprime numbers as $6 = 3 \cdot 2$, the Euler's Totient function can be expressed as the product of the Euler's Totient function of each factor [35]. As all the combinations of a_s in any stage s are obtained as the product of the Euler's Totient function of each factor, the number of possible PFA solutions is obtained as the Euler's Totient function of the factorization, i.e.,

$$\text{n}^\text{o} \text{ solutions} = \prod_{s=1}^{M} \varphi(N_s) = \varphi(N). \tag{36}$$

B. Connecting Stages

Let us consider the factorization $N = N_1 \cdot N_2 \cdot \ldots \cdot N_M$ and define the data index of any certain stage with the digit representation $d_1 d_2 \ldots d_M$ [5]. Each digit is associated with its corresponding stage. Thus, the values that a certain d_s can have are the residues of N_s. This is $d_s = 0, 1, \ldots, N_s - 1$.

From P2 FFTs, we know that butterflies at any stage s operate on data whose index differs only in the bit associated with this stage [4]. This corresponds to the digit d_s according to the notation that we are considering. We also know that consecutive stages connect data with the same index [5]. Thus, the wire of the flow graph that has any specific values of $d_1 d_2 \ldots d_M$ at any stage s must be connected with the wire with the same values $d_1 d_2 \ldots d_M$ at stage $s + 1$. Finally, the digits $d_1 d_2 \ldots d_M$ can be sorted in any order. By creating an analogy with a counter, we consider that the LSB is the digit that changes first between consecutive inputs of the flow graph and the MSB is the index that changes last. The proposed approach is based on these three principles.

Table I shows the procedure to obtain the connections between the butterflies for a 30-point FFT whose factorization is $5 \cdot 3 \cdot 2$. The digit representation of the index is $d_1 d_2 d_3$. Then, $d_1 = \{0, 1, 2, 3, 4\}$, $d_2 = \{0, 1, 2\}$ and $d_3 = \{0, 1\}$.

In the problem statement described in Section III, we assume that butterflies at any stage s operate on consecutive edges from top to bottom in the flow graph. As the digit in the LSB position is the digit that changes between consecutive edges and butterflies at stage s operate with data whose index differ in d_s, then at any stage d_s must be placed in the LSB position. According to this, in Table I it can be observed that d_1 is the LSB in stage $s = 1$, and d_2 and d_3 are the LSBs in stages 2 and 3, respectively. The order of the other two digits at each stage can be chosen arbitrarily. Thus, in the example of Table I each stage has two possible orders for the digits. These are $d_3 d_2 d_1$ and $d_2 d_3 d_1$ for the first stage, $d_1 d_3 d_2$ and $d_3 d_1 d_2$ for the second stage, and $d_2 d_1 d_3$ and $d_1 d_2 d_3$ for the third stage. By combining all the alternatives, there is a total of eight alternatives for this factorization. In the example of Table I, we have considered the orders $d_2 d_3 d_1$, $d_1 d_3 d_2$, and $d_1 d_2 d_3$ for stages 1 to 3, respectively.

The next step to obtain the permutations in the flow graph is to connect the values with the same index between consecutive

TABLE I
PROCEDURE TO CONNECT DATA BETWEEN STAGES. EXAMPLE OF A 30-POINT FFT. PERMUTATION $d_2d_3d_1$ ($s=1$), $d_1d_3d_2$ ($s=2$) AND $d_1d_2d_3$ ($s=3$).

$s=1$	$s=2$	$s=3$
$d_2d_3d_1$	$d_1d_3d_2$	$d_1d_2d_3$
000	000	000
001	001	001
002	002	010
003	010	011
004	011	020
010	012	021
011	100	100
012	101	101
013	102	110
014	110	111
100	111	120
101	112	121
102	200	200
103	201	201
104	202	210
110	210	211
111	211	220
112	212	221
113	300	300
114	301	301
200	302	310
201	310	311
202	311	320
203	312	321
204	400	400
210	401	401
211	402	410
212	410	411
213	411	420
214	412	421
$d_2d_3d_1$	$d_1d_3d_2$	$d_1d_2d_3$
$\searrow\!\!\searrow$	$\searrow$	
$d_1d_2d_3$	$d_1d_2d_3$	$d_1d_2d_3$
000	000	000
100	010	001
200	020	010
300	001	011
400	011	020
001	021	021
101	100	100
201	110	101
301	120	110
401	101	111
010	111	120
110	121	121
210	200	200
310	210	201
410	220	210
011	201	211
111	211	220
211	221	221
311	300	300
411	310	301
020	320	310
120	301	311
220	311	320
320	321	321
420	400	400
021	410	401
121	420	410
221	401	411
321	411	420
421	421	421

stages. This can be carried out directly from the upper part of Table I by observing the value that each digit takes in consecutive stages. For instance, the 11th data index in stage 1 and the 2nd one in stage 2 have $d_3=0$, $d_2=1$, and $d_1=0$, so they must be connected together. Note that it may be confusing to see which values from consecutive stages are the same since the digits are in different order. To make the connections easier, we recommend to rewrite the digits following the same order. This is what is done at the bottom of Table I. There, the digits at all the stages are ordered as $d_1d_2d_3$. Note that this representation loses the information about which bit varies first, so it can only be used for the purpose of connecting the stages. Now that all the digits are ordered as $d_1d_2d_3$, it is easy to identify which indexes must be connected between consecutive stages, which is done in this bottom part of the figure.

The proposed approach can be easily generalized for the case of an N-point FFT, with $N=N_1\cdot N_2\cdot\ldots\cdot N_M$. As the selection of the digit order at a certain stage s only requires that the least significant digit is d_s, there is room to choose the order of the remaining $M-1$ digits. As a consequence, the number of possible orders of the digits at any stage is equal to the number of permutations of these $M-1$ digits, i.e.,

$$\text{nº permutations}_{\text{1 stage}}=(M-1)!, \tag{37}$$

which involves the factorial operator. Considering all the stages, the total number of permutations given M factors is

$$\text{nº permutations}=(M-1)!^M. \tag{38}$$

C. Input and Output Order

In Section IV-B, the digit representation with the notation $d_1d_2\ldots d_M$ is extracted. This is a general notation for all the stages. However, the model of the index mapping considers the notation n_s and k_s for the input digits and output digits, respectively. To get the decimal values of n in the flow graph, let us identify n_s as d_s when $s=1$. To get the decimal values of k in the flow graph, let us identify k_s as d_s when $s=M$. Then, equations (35a) and (35b) include the necessary information to extract the decimal values of n_s and k_s, taking into account that the order in which n_s and k_s must increase depends on the order of the digits $d_1d_2\ldots d_M$ at the first and last stage, respectively.

Fig. 4 shows one possible flow graph of a 30-point FFT whose factorization is $5\cdot3\cdot2$ using the proposed methodology. In order to obtain the coefficients, $\{a_1,a_2,a_3\}=\{3,1,1\}$ has been chosen. Thus, the index mapping according to (35a) and (35b) becomes

$$n\equiv 18\cdot n_1+10\cdot n_2+15\cdot n_3 \pmod{30}, \tag{39a}$$
$$k\equiv 12\cdot k_1+10\cdot k_2+15\cdot k_3 \pmod{30}. \tag{39b}$$

The digit orders have been chosen as $d_2d_3d_1$, $d_1d_3d_2$ and $d_1d_2d_3$ for stages 1 to 3, respectively, which corresponds to the case in Table I. Therefore, for stage $s=1$ the digits in terms of n_s are $n_2n_3n_1$. This means that n_1 is the LSB and changes first. Then n_3 changes and, finally, n_2. As n_1 is the LSB and its weight is 18 according to (39a), the index n of

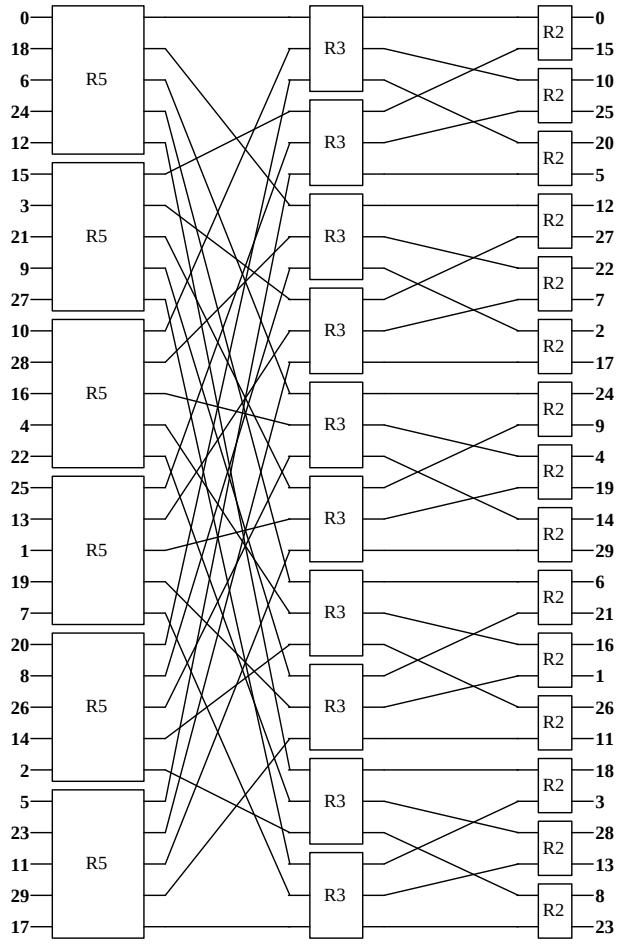

Fig. 4. Flow graph of a 30-point FFT with factorization $5 \cdot 3 \cdot 2$.

the second input in Fig. 4 is 18. The digit n_1 takes the values 0 to 4, leading to the input indexes of the upper butterfly at stage 1. Then, n_3 starts to change, being $n_3 = 1$ for all the inputs of the second butterfly. In the third butterfly, n_3 is zero again, as it only takes the values 0 and 1, whereas $n_2 = 1$. Thus, the index of the first input of the third butterfly is equal to the weight of n_2, as both n_1 and n_3 are equal to 0 for this edge.

The order of the outputs can be obtained analogously, by taking into account the weights in equation (39b) and the fact that the digits at the third stage are ordered as $d_1 d_2 d_3$, so the order of the digits of the output index is $k_1 k_2 k_3$.

Finally, the permutations between FFT stages are obtained from the order of the digits $d_1 d_2 \dots d_M$ at the stages that are connected, as explained in Section IV-B. In the particular example of Fig. 4, the orders and, therefore, the connections are the same as in Table I. The outputs from the first two radix-5 butterflies are connected to the first input of the radix-3 butterflies, whereas the outputs of the next two radix-5 butterflies are connected to the second inputs of the radix-3 butterflies. This pattern continues, maintaining structured connections between the radix-5 and radix-3 butterflies. Similarly, the outputs from the radix-3 butterflies are connected to the radix-2 butterflies, following a systematic distribution: The outputs of the odd radix-3 butterflies are connected to the first input of the radix-2 butterflies and the outputs of the even butterflies are connected to the second input of the radix-2 butterflies.

As a result, the proposed approach obtains the FFT flow graph in a systematic way, eliminates the twiddle factors between butterflies, and the solution uses standard radix-r butterflies that calculate an r-point FFT.

TABLE II
POSSIBLE PFA INDEX MAPPINGS FOR A 30-POINT FFT WITH THE FACTORIZATION $5 \cdot 3 \cdot 2$. FORMULATION OF n AND k EVALUATED MODULO 30.

$\{a_1, a_2, a_3\}$	n	k
$\{1,1,1\}$	$6n_1 + 10n_2 + 15n_3$	$6k_1 + 10k_2 + 15k_3$
$\{1,2,1\}$	$6n_1 + 20n_2 + 15n_3$	$6k_1 + 20k_2 + 15k_3$
$\{2,1,1\}$	$12n_1 + 10n_2 + 15n_3$	$18k_1 + 10k_2 + 15k_3$
$\{2,2,1\}$	$12n_1 + 20n_2 + 15n_3$	$18k_1 + 20k_2 + 15k_3$
$\{3,1,1\}$	$18n_1 + 10n_2 + 15n_3$	$12k_1 + 10k_2 + 15k_3$
$\{3,2,1\}$	$18n_1 + 20n_2 + 15n_3$	$12k_1 + 20k_2 + 15k_3$
$\{4,1,1\}$	$24n_1 + 10n_2 + 15n_3$	$24k_1 + 10k_2 + 15k_3$
$\{4,2,1\}$	$24n_1 + 20n_2 + 15n_3$	$24k_1 + 20k_2 + 15k_3$

D. Obtaining All the Flow Graphs

The number of the proposed PFA solutions is stated in (36) and the number of possible permutations is stated in (38). The first subproblem obtains all the possible input and output orders by a given connection of the butterflies. The second subproblem changes the connections of the butterflies independently of the set of coefficients. This problem can be understood as changing the placement of the butterflies in the flow graph. However, even when the order of the input and output data may change due to the new placement of the butterflies from top to bottom in the flow graph, the algorithm itself remains the same, as the equations (35a) and (35b) do not change. Likewise, changing the order of the butterflies can not lead to new results for equations (35a) and (35b). As a result, each subproblem is independent of the other and the total number of possible flow graphs is the product of equations (36) and (38), i.e.,

$$\text{n}^{\text{o}} \text{ representations}_{1 \text{ factorization}} = (M-1)!^M \cdot \varphi(N). \quad (40)$$

By following the example of the 30-point FFT, there are 8 different PFA solutions and 8 possible permutations. This means 64 possible flow graphs for just the factorization $5 \cdot 3 \cdot 2$.

Table II shows the 8 different index mappings that lead to the 8 different algorithms for the example of the 30-point FFT. The values of α_s and β_s are calculated according the values of a_s. Note that the case $\{3,1,1\}$ leads to the flow graph of Fig. 4.

Table III shows the different digit orders that form the counter at each stage for the 30-point FFT. There are 8 possible combinations that can be chosen. Each combination leads to different connections between butterflies of consecutive stages. Note that Fig. 4 implements the permutations of the first combination. Note also that, for any stage s, the last digit

TABLE III
POSSIBLE PERMUTATIONS FOR A FACTORIZATION THAT CONSISTS OF THREE FACTORS.

Stage 1	Stage 2	Stage 3
$d_2d_3d_1$	$d_1d_3d_2$	$d_1d_2d_3$
$d_2d_3d_1$	$d_1d_3d_2$	$d_2d_1d_3$
$d_2d_3d_1$	$d_3d_1d_2$	$d_1d_2d_3$
$d_2d_3d_1$	$d_3d_1d_2$	$d_2d_1d_3$
$d_3d_2d_1$	$d_1d_3d_2$	$d_1d_2d_3$
$d_3d_2d_1$	$d_1d_3d_2$	$d_2d_1d_3$
$d_3d_2d_1$	$d_3d_1d_2$	$d_1d_2d_3$
$d_3d_2d_1$	$d_3d_1d_2$	$d_2d_1d_3$

is always d_s in any combination. This ensures that inputs of the same butterfly share the same digit representation with the exception of the digit d_s. Any combination of rows of Table II with any row of Table III leads to one possible flow graph out of the 64 possible representations.

It is important to highlight that the index mappings proposed in Table II only work for the specific factorization $5\cdot3\cdot2$. Any other order of the factors will lead to other index mappings. Likewise, by changing the order of the factors, the digits $d_1d_2d_3$ will take a different range of values. For instance, for the factorization $3\cdot2\cdot5$, it is fulfilled that $N_1=3$, $N_2=2$ and $N_3=5$. Thus, the range of the digits becomes $d_1=\{0,1,2\}$, $d_2=\{0,1\}$ and $d_3=\{0,1,2,3,4\}$. As a result, the total number of representations for a given FFT size N split into M factors provided in (40) must be multiplied by all the permutations of the M factors of N, which is equal to the factorial of M, leading to

$$\text{n}^\text{o}\text{ representations} = (M-1)!^M \cdot \varphi(N) \cdot M! \tag{41}$$

E. Decomposing Any FFT Size

When the factorization of N includes some factors that are not coprime, the prime factor algorithm can be combined with the Cooley-Tukey algorithm to carry out the decomposition. This is more likely to occur for large values of N, where the factorization of N into prime factors may include several instances of the same prime number.

There are different ways to group factors. The conventional method [26], [37]–[39] is to decompose N first in terms of powers of different prime numbers using PFA and then use the Cooley-Tukey algorithm to decompose each power of a prime number. This way, for a 1500-point FFT, N is decomposed first as $N=N_1\cdot N_2\cdot N_3=125\cdot4\cdot3$ using PFA and then Cooley-Tukey is used to decompose $N_1=125$ as $5\cdot5\cdot5$ and $N_2=4$ as $2\cdot2$. This method ensures that the PFA factorization always generates coprime FFT sizes, as these sizes are powers of different prime numbers.

Contrary to previous approaches, in this work we suggest starting the decomposition of N with the Cooley-Tukey algorithm to achieve numbers whose factors are coprime and can be decomposed later using PFA. This idea has already been used in our recent work [40]. If we consider the previous example for $N=1500$, it can be decomposed into $N=30\cdot10\cdot5$, being the 30-point FFT and the 10-point FFT designed with PFA.

It can be observed that both approaches lead to a different number of stages with rotations. Note that rotations appear only when the Cooley-Tukey algorithm is applied and it introduces one set of rotations in between each pair of stages that it creates. Thus, for any number

$$N = N_1^{q_1}\cdot N_2^{q_2}\dots N_M^{q_M}, \tag{42}$$

where each N_i, $i=1,\dots,M$, is a prime number, the approach in [26], [37]–[39] requires a number of stages with rotations equal to

$$\text{Rot. stages } = \sum_{i=1}^{M}(q_i-1). \tag{43}$$

This means that each increment in the exponent of any factor beyond 1 adds one stage of rotations. This occurs because the Cooley-Tukey algorithm decomposes each sub-FFT consisting of a power of a prime number.

For the same decomposition in (42), the proposed approach requires a number of stages with rotations equal to

$$\text{Rot. stages } = \max_i\{q_i-1\}. \tag{44}$$

In this case, the Cooley-Tukey creates a number of sub-FFTs equal to the maximum exponent among all q_i, including one stage of rotations in between each pair of consecutive stages. Later, the PFA is applied to the sub-FFTs without adding any rotation between stages.

By comparing (43) and (44) it can be noted that the proposed approach always results in an equal or smaller number of stages with rotations.

To show mathematically how to combine the Cooley-Tukey algorithm with the proposed prime factor algorithm, let us consider the example of a 1500-point FFT. First, we can decompose N using the Cooley-Tukey algorithm as $N=30\cdot10\cdot5$, being the FFT divided into three sub-FFTs. As each sub-FFT consists of coprime factors, we can design each of them using the proposed prime factor algorithm. According to (35a) and (35b), and choosing $a_1=1$, $a_2=1$ and $a_3=1$, the index mapping of the 30-point FFT is

$$n_{30} \equiv 6\cdot n_{30,1} + 10\cdot n_{30,2} + 15\cdot n_{30,3} \pmod{30}, \tag{45a}$$
$$k_{30} \equiv 6\cdot k_{30,1} + 10\cdot k_{30,2} + 15\cdot k_{30,3} \pmod{30}. \tag{45b}$$

Likewise, choosing $a_1=1$, $a_2=1$, the input and output formula for the 10-point FFT is

$$n_{10} \equiv 6\cdot n_{10,1} + 5\cdot n_{10,2} \pmod{10}, \tag{46a}$$
$$k_{10} \equiv 2\cdot k_{10,1} + 5\cdot k_{10,2} \pmod{10}. \tag{46b}$$

Based on the coefficients of the Cooley-Tukey index mapping in (3a) and (3b), we can extract the final index mapping of the 1500-point FFT by combining the 30-point, 10-point, and 5-point FFTs as

$$n_{1500} \equiv 50\cdot n_{30} + 5\cdot n_{10} + n_5 \pmod{1500}, \tag{47a}$$
$$k_{1500} \equiv k_{30} + 30\cdot k_{10} + 300\cdot k_5 \pmod{1500}. \tag{47b}$$

Note that n_{30}, k_{30} are the residues of the 30-point FFT, n_{10}, k_{10} are the residues of the 10-point FFT and the n_5, k_5 are the residues of the 5-point FFT. Thus, each n_s, k_s is evaluated modulo N_s.

TABLE IV
COMPARISON BETWEEN THE PROPOSED AND PREVIOUS PRIME FACTOR ALGORITHMS.

Figures of merit	Cooley-Tukey [1]	Good [12]	Burrus [13]	Temperton [14]	Lun [15]	Prop.
Twiddle factors	yes	no	no	no	no	no
Homogeneous butterflies	yes	yes	no	no	yes	yes
Any factorization	yes	no	no	no	no	no
Relationship between factors	Any	Coprime	Coprime	Coprime	Quadratic residues	Coprime
Number of factors	Any	Any	Any	Any	Any	Any
Number of I/O orders	1	1	$\varphi(N)$	1	$2^{\bar{M}}$	$\varphi(N)$
Internal permutations	-	-	-	-	-	$(M-1)!^M$
Obtains the flow graph	no	no	no	no	no	yes

TABLE V
NUMBER OF OPERATIONS REQUIRED IN DIFFERENT FFTS USING DIFFERENT ALGORITHMS WITH THE SAME BUTTERFLIES.

		Rader [6]		Winograd [7]		Cooley-Tukey [1]			PFA [12]–[14]			Lun [15]			Prop.		
N	Factors	Mult.	Add.	Mult.	Add.	Ord.	Mult.	Add.	Ord.	Mult.	Add.	Ord.	Mult.	Add.	Ord.	Mult.	Add.
2	2	-	-	-	-	-	0	4	-	-	-	-	-	-	-	-	-
3	3	2	12	4	12	-	-	-	-	-	-	-	-	-	-	-	-
4	4	-	-	-	-	-	0	17	-	-	-	-	-	-	-	-	-
5	5	10	36	8	34	-	-	-	-	-	-	-	-	-	-	-	-
6	$3 \cdot 2$	-	-	-	-	1	10	42	1	4	36	-	-	-	2	4	36
8	$2 \cdot 2^2$	-	-	-	-	1	6	57	-	-	-	-	-	-	-	-	-
10	$5 \cdot 2$	-	-	-	-	1	28	100	1	16	88	-	-	-	4	16	88
12	$4 \cdot 3$	-	-	-	-	1	20	113	1	8	99	-	-	-	4	8	99
15	$5 \cdot 3$	-	-	-	-	1	58	186	1	34	162	-	-	-	8	34	162
16	$2^2 \cdot 2^2$	-	-	-	-	1	24	161	-	-	-	-	-	-	-	-	-
20	$5 \cdot 4$	-	-	-	-	1	68	257	1	32	221	4	32	221	8	32	221
30	$5 \cdot 3 \cdot 2$	-	-	-	-	1	155	472	1	68	384	4	68	384	8	68	384
32	$2 \cdot 2^2 \cdot 2^2$	-	-	-	-	1	84	425	-	-	-	-	-	-	-	-	-
60	$5 \cdot 3 \cdot 4$	-	-	-	-	1	322	1101	1	136	903	-	-	-	16	136	903
64	$2^2 \cdot 2^2 \cdot 2^2$	-	-	-	-	1	228	1049	-	-	-	-	-	-	-	-	-

With respect to the connection between stages, the methodology proposed in Section IV-B to obtain the flow graph is applicable to both PFA and Cooley-Tukey algorithms. Therefore, by combining the connection between stages in Section IV-B and the index mapping discussed in this section, the proposed algorithm and the Cooley-Tukey algorithm can be used together to decompose an FFT of any size N.

V. COMPARISON

Table IV compares the proposed prime factor algorithm with previous prime factor algorithms [12]–[15] and the Cooley-Tukey algorithm [1]. The figures of merit include qualitative parameters with respect to the flow graphs and quantitative parameters with respect to the index mapping.

The proposed algorithm and previous prime factor algorithms do not include twiddle factors between stages. Conversely, the Cooley-Tukey algorithm has twiddle factors between stages, which makes the flow graph include complex multiplications as in Fig. 1. From the algorithms that do not include twiddle factors, [12], [15], and the proposed one use homogeneous butterflies. Thus, the same design of a butterfly is used for any factorization that requires it. The other algorithms need several designs for the same butterfly size with different rotations each.

All the prime factors algorithms shown in Table IV can not be used for any arbitrary factorization as does the Cooley-Tukey algorithm [1]. In the proposed algorithm and in [12]–[14], each factor has to be coprime to each other. This is the payoff to eliminate the twiddle factors of [1]. In [15], the relation between the factors is even more strict, as they have to be quadratic residues to each other. This restriction appears because the index mapping involves square roots of numbers evaluated under a modulo operator, and these values must exist. Regarding the the number of factors in the factorization there is no restriction in any of the algorithms.

The index mappings proposed in [1], [12], [14] lead to a direct input and output order. The work [15] expands the number of different input and output orders to $2^{\bar{M}}$, where $\bar{M}$ is the number of factors greater than two in the factorization. As each stage involves a square root, two different solutions can be chosen, with the exception of a stage whose size is two. The square root of an odd number modulo 2 is always the unity, which involves only one solution. In the proposed algorithm and [13] additional solutions can be chosen. Though both algorithms present the same number of solutions for the I/O order, in [13] the input and output orders are different because they use different, non homogeneous butterflies, whereas the proposed approach uses standard homogeneous butterflies..

The proposed work includes the methodology to extract

different internal permutations between butterflies, whereas in previous works this is not addressed. Finally, the proposed approach explains how to obtain the FFT flow graph of any NP2 FFT and calculates how many flow graphs exist for any FFT size, which has not been considered before in the literature.

Table V shows the number of operations required in FFTs with different sizes using the Cooley-Tukey algorithm [1], the prime factors algorithms in [12]–[14], the prime factor algorithm in [15], and the proposed prime factor algorithm. The figures of merit are the number of different input and output orders (Ord.), which corresponds to the different index mappings, the number of real multiplications (Mult.), and the number of real additions/subtractions (Add).

The comparison assumes that the factorization maintains the factors in the same order and the butterflies are homogeneous, resulting in a different number of I/O orders for [13] in Table V with respect to Table IV. Regarding operations, the comparison assumes that the multiplications by powers of two do not require any hardware, trivial rotations count as one real addition, and general rotations count as three real multiplications and three real additions, according to [41]. For the PFA algorithms in the last three columns of the table, radix-2 and radix-4 butterflies use the Cooley-Tukey algorithm, radix-3 butterflies use Rader's algorithm [6], and radix-5 butterflies use Winograd's algorithm [7]. The operations of these butterflies are reported in columns 3, 4, and 5 of Table V for N = 2, 4, 3, and 5.

Regarding the FFT sizes, the Cooley-Tukey algorithm can be applied to every N reported in the table, except for the radix-3 and radix-5 butterflies. This covers both P2 and NP2 sizes. Power-of-two FFT sizes are not covered by any prime factor algorithm, as they need the factors of N to be coprime. In fact, the PFA in [15] can only be used when $N = 20$ and $N = 30$ as the factors need to be quadratic residues.

Regarding the operations, the proposed approach and the rest of the prime factor algorithms reduce both the number of real multiplications and the number of real additions compared to the Cooley-Tukey algorithm for each NP2 size. This reduction comes from removing the twiddle factors that must be calculated in the Cooley-Tukey algorithm. The difference between the proposed algorithm and the rest of the prime factor algorithms is the number of index mappings that can be used for the same factorization, as reported in Table IV. As the butterflies are assumed to be standard in Table V, the number of input and output orders of [13] is reduced to only one as in [12], [14], whereas the number of input and output orders of the proposed algorithm remains $\varphi(N)$. Additionally, the value of $\varphi(N)$ will always be higher than the number of input and output orders obtained by [15]. This is demonstrated by knowing that the function 2^M increases by two for each factor greater than two in the factorization, whereas $\varphi(N)$ increases by a factor $\varphi(N_s)$ for each factor in the factorization. For values of N_s greater than two, the minimum value that $\varphi(N_s)$ can have is 2. This means that the number of input and output orders would be the same only for $N = 6$ and $N = 12$. However, these factors are not considered by [15]. As a result, the proposed algorithm achieves the maximum number of different index mappings.

VI. Implementation and Experimental Results

In order to demonstrate the capabilities of the proposed algorithm, FFTs of sizes $N = 6$, 15, 30 and 60 have been implemented in hardware using the proposed prime factor algorithm. The fully parallel architecture [31], [33] has been chosen for the proposed implementations as it is a direct implementation of the operations calculated in the flow graph. This way, one real addition in the flow graph is translated into one adder in hardware. Analogously, one real multiplication in the flow graph is translated into a multiplier in hardware. Thus, a reduction of operations using the proposed algorithm is directly translated into a reduction of area. As an example, the flow graph in Fig. 4 is directly the fully parallel implementation of a 30-point FFT using the proposed algorithm. Note that it consists of butterflies and intermediate connections. The butterflies that are used in the proposed architectures are the parallel ones reported in [9]. Therefore, radix-2 and radix-4 butterflies are designed using the Cooley-Tukey algorithm, the radix-3 butterfly is designed with the Rader's algorithm and the radix-5 butterfly is designed using the Winograd's algorithm. The number of operations that each butterfly requires is reported in Table V for $N = 2$, 3, 4, and 5, respectively. The number of operations required for each proposed FFT is reported in Table V for $N = 6$, 15, 30, and 60. The intermediate connections do not require any hardware cost in fully parallel implementations as all data are processed in parallel. Therefore, the connections are carried out with wires. Also, FFT inputs and outputs are ordered according to its index mapping. Therefore, if external data are required in natural order, inputs and outputs are hard-wired following the requirement with no additional cost due to the ordering process. It should be mentioned that if any architecture with parallelization less than N were used, additional circuitry for reordering data would be needed, adding hardware cost to the design.

Table VI shows the experimental results of the proposed designs (Prop.) and compares them to state-of-the-art fully parallel FFT implementations [31], [33]. Experimental results for the proposed approach are provided for Virtex UltraScale XCVU190-FLGA2577-2-E and Virtex UltraScale+ XCVU37P-FSVH2892-2L-E. The work [33] provides the experimental results on a Virtex UltraScale+ XCVU35P-FSVH2104-3-E, and the work [31] provides the experimental results on a Virtex UltraScale XCVU190-FLGA2577-2-E. In Table VI, VU stands for Virtex UltraScale and VU+ stands for Virtex UltraScale+. For a fair comparison, the proposed architectures have been designed under the same conditions as [31]: The inputs and outputs are registered, the rotations calculated inside the butterflies are carried out with shit-and-add operations, pipelining is applied after each adder, and a word length of 16 bits for the real part and 16 bits for the imaginary part is used. No scaling between stages is applied, which means that the input word length is the same as the output word length.

TABLE VI
COMPARISON BETWEEN THE PROPOSED AND PREVIOUS PRIME FACTOR ALGORITHMS.

N	Ref.	Arch. (⋆)	FPGA Device	WL (bits)	Area (‡) LUT	FF	CARRY8	CLB	BRAM	DSP	f_{CLK} (MHz)	Latency (cyc.)	(ns)	Th. (GS/s)	Power (W)	SQNR (dB)
6	Prop.	FP	VU+	16	945	1418	148	199	0	0	850	7	8	5.10	0.12	75.6
6	Prop.	FP	VU	16	945	1418	148	215	0	0	750	7	9	4.50	0.12	75.6
8	[31]	FP	VU	16	1380	1636	NP	272	0	0	742	7	9	5.93	0.33	80.2
15	Prop.	FP	VU+	16	5091	7455	787	1027	0	0	750	13	17	11.25	0.70	66.8
15	Prop.	FP	VU	16	5087	7455	787	1067	0	0	650	13	20	9.75	0.68	66.8
15	[33]	FP	VU+	18	8000	19400	NP	NP	0	264	724	NP	NP	10.86	NP	63.0
16	[33]	FP	VU+	16	8300	22800	NP	NP	0	0	891	NP	NP	14.25	NP	75.0
16	[31]	FP	VU	16	4376	5586	NP	801	0	0	636	8	13	10.17	0.60	75.9
30	Prop.	FP	VU+	16	11141	15870	1754	2342	0	0	750	14	19	22.50	1.71	64.8
30	Prop.	FP	VU	16	11139	15904	1754	2409	0	0	600	14	23	18.00	1.57	64.8
30	[33]	FP	VU+	18	25100	49500	NP	NP	0	694	648	NP	NP	19.44	NP	66.0
32	[33]	FP	VU+	16	22900	62000	NP	NP	0	0	891	NP	NP	28.51	NP	65.0
32	[31]	FP	VU	16	13317	19412	NP	2543	0	0	655	15	23	20.96	1.91	72.7
60	Prop.	FP	VU+	16	24321	33775	3868	5216	0	0	650	15	23	39.00	3.14	62.9
60	Prop.	FP	VU	16	24320	33799	3868	5214	0	0	550	15	25	36.00	3.09	62.9
64	[33]	FP	VU+	16	59200	166300	NP	NP	0	0	794	NP	NP	50.08	NP	54.0
64	[31]	FP	VU	16	35057	51414	NP	6594	0	0	579	16	28	37.06	4.65	69.4

(⋆): FP stands for fully parallel architecture. (‡): NP stands for not provided.

The figures of merit of Table VI are area (LUT, FF, CARRY8, CLB, BRAM, and DSP), maximum clock frequency (f_{CLK}), latency, throughput (Th.), dynamic power consumption, and signal-to-quantization-noise ratio (SQNR) [42], [43]. The proposed power results have been measured using Vivado after simulating 1000 FFTs, extracting the activity factors for accurate estimation. The SQNR is measured according to [43], using the previous 1000 FFTs for testing.

Comparing the proposed 6-point FFT with the 8-point FFT in [31], the proposed design achieves a notable reduction in area. The number of LUTs is reduced from 1380 to 945, which corresponds to a 31% decrease, even though $N = 6$ is only 25% smaller than $N = 8$. Similarly, the number of FFs decreases from 1636 to 1418, further highlighting the efficiency in resource usage. Both designs have an identical latency. Regarding power consumption, the proposed design presents a significant reduction, reducing from 0.33 W to 0.12 W, which represents a 64% decrease.

Comparing the proposed 15-point FFT with the 16-point FFT in [31], the proposed design shows a moderate increase in area. The number of LUTs increases from 4376 to 5087, which represents a 16% increase. The number of FFs also increases from 5586 to 7455, reflecting the higher number of operations that are needed in the 15-point FFT, according to Table V. In terms of latency, the proposed design requires more cycles, increasing from 8 to 13 cycles. Similarly, power consumption slightly increases, from 0.60 W to 0.68 W, which represents a 13% increase, making the 15-point FFT less efficient than the 16-point one, as expected from the number of operations reported in Table V for these sizes. However, the proposed 15-point FFT notably reduces area compared with the previous 16-point FFT reported in [33]. This is due to the fact that the 16-point FFT in [31] uses radix-2^2 algorithm whereas the 16-point FFT in [33] uses radix-2 algorithm, increasing the number of non-trivial rotations. The proposed 15-point FFT significantly reduces area compared with the previous 15-point FFT in [33]. The proposed design presents a 36% and 61% reduction of LUTs and FFs, respectively, while none of the 264 DSPs in [33] are used in the proposed design. Despite using two fewer bits in the proposed design, the resulting area reduction is obtained by removing the twiddle factors used in [33].

Comparing the proposed 30-point FFT with the 32-point FFT in [31], the number of LUTs is reduced from 13,317 to 11,139, which represents a 16% reduction, and FFs are reduced by 18% from 19412 to 15904, showing a marked improvement in resource efficiency. In terms of latency, the proposed design reduces the latency by one clock cycle. Regarding power consumption, the proposed design consumes 1.57 W, which is 18% less than the 1.91 W consumed by the 32-point FFT, making the proposed architecture more resource-efficient despite being only 6% smaller in size. Compared with the 32-point FFT in [33], the proposed 30-point FFT reduces by 51% the number of LUTs and 74% of the FFs. The area improvement is even bigger regarding the previous 30-point FFT in [33], where 694 DSPs are eliminated in the proposed design while reducing 55% and 68% LUTs and FFs, respectively. Despite using two fewer bits in the proposed design, the resulting area reduction is obtained by removing the twiddle factors used in [33].

Comparing the proposed 60-point FFT with the 64-point FFT in [31], the proposed design significantly reduces area. The number of LUTs is reduced from 35057 to 24320, corresponding to a 30% reduction, and the number of FFs decreases by 34% from 51414 to 33799, making the proposed design much more efficient in terms of area usage. Regarding

latency, the proposed design reduces the latency by one clock cycle. In terms of power consumption, the proposed design shows a substantial reduction, consuming 3.09 W compared to 4.65 W in the 64-point FFT in [31], which corresponds to a 33% reduction of power consumption. Compared to the radix-2 64-point FFT in [33], the number of LUTs is reduced from 59200 to 24321 in the proposed design, which corresponds to a 59% reduction, and the FFs are reduced from 166300 to 33775, which corresponds to an 80% reduction. Being N only 6% smaller, this case highlights the potential of the NP2 FFTs compared to efficient radix-2^2 and radix-2 power-of-two FFTs.

All the proposed architectures implemented on the Virtex UltraScale (VU) achieve similar clock frequencies as the power-of-two architectures in [31]. This fact emphasizes the degree of optimization of the architectures, even though the throughput is slightly smaller, as N is smaller. The architectures implemented on the Virtex UltraScale+ achieve higher clock frequencies, and the throughput also increases for the proposed architectures and the architectures in [33]. Regarding SQNR, the proposed NP2 FFTs present smaller SQNR than the P2 architectures reported in [31]. This fact comes from the precision lost by truncating more bits in the radix-3 and radix-5 butterflies after each stage. Still, the SQNR is improved compared to the architectures presented in [33].

VII. Conclusions

This work has presented a methodology to obtain the flow graph of an NP2 FFT given its factorization. The methodology is divided into two independent subproblems. First, a new prime factor algorithm is derived to get the coefficients of the linear form of the index mapping. Second, the connections between the butterflies in the flow graph are derived. Both subproblems offer several solutions that have been quantified. The combination of these solutions leads to different flow graphs.

The proposed approach not only eliminates the need for twiddle factors between butterflies, as in previous PFAs, but also works for any factorization of coprime numbers and uses standard butterflies. Furthermore, the proposed algorithm explains how to carry out the internal permutations in the flow graph, which was not considered in previous works. Additionally, the proposed approach can be further combined with the Cooley-Tukey algorithm to decompose any FFT size efficiently. As a result, this work solves the challenge of representing NP2 FFT flow graphs, which is the basis for deriving efficient NP2 FFT hardware architectures in the future.

The analysis of the number of operations in the proposed NP2 FFTs reveals a reduction in operations compared to their P2 homologous, especially for sizes such as $N = 30$ and $N = 60$. Experimental results on FPGAs show that some NP2 FFTs of sizes close to a P2 size require significantly fewer hardware resources and less power consumption than the corresponding P2 FFTs. This demonstrates that NP2 FFTs can be even more efficient than P2 ones, which is a very relevant conclusion taking into account that, so far, most of the FFT research dealt with P2 sizes. This opens the research field towards exploring NP2 FFTs in depth.

References

[1] 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.

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

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

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

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

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

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

[8] F. Qureshi, M. Garrido, and O. Gustafsson, "Unified architecture for 2, 3, 4, 5, and 7-point DFTs based on winograd fourier transform algorithm," *Electron. Lett.*, vol. 49, no. 5, pp. 348–345, Feb. 2013.

[9] V. M. Bautista, M. Garrido, and M. López-Vallejo, "Serial butterflies for non-power-of-two FFT architectures in 5G and beyond," *IEEE Trans. Circuits Syst. I*, vol. 70, no. 10, pp. 3992–4003, Oct. 2023.

[10] V. M. Bautista and M. Garrido, "An automatic generator of non-power-of-two SDF FFT architectures for 5G and beyond," in *Proc. Conf. Design Circuits Integr. Syst.*, Nov. 2023, pp. 1–6.

[11] J. Löfgren and P. Nilsson, "On hardware implementation of radix-3 and radix-5 FFT kernels for LTE systems," in *Proc. NORCHIP*, Nov. 2011, pp. 1–4.

[12] I. Good, "The relationship between two fast fourier transforms," *IEEE Trans. Comput.*, vol. C-20, no. 3, pp. 310–317, Mar. 1971.

[13] C. Burrus and P. Eschenbacher, "An in-place, in-order prime factor FFT algorithms," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 29, no. 4, pp. 806–817, Aug. 1981.

[14] C. Temperton, "Implementation of a self-sorting in-place prime factor FFT algorithm," *J. Comput. Physics*, vol. 58, no. 3, pp. 283–299, May 1985.

[15] D. P. K. Lun and W. C. Siu, "An analysis for the realization of an in-place and in-order prime factor algorithm," *IEEE Trans. Signal Process.*, vol. 41, no. 7, pp. 2362–2370, Jul. 1993.

[16] J. Schatzman, "Index mappings for the fast fourier transform," *IEEE Trans. Signal Process.*, vol. 44, no. 3, pp. 717–719, Mar. 1996.

[17] 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.*, May 2019, pp. 1–5.

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

[19] 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.

[20] 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.

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

[22] G. Wang, B. Yin, I. Cho, J. R. Cavallaro, S. Bhattacharyya, and J. Takala, "Efficient architecture mapping of FFT/IFFT for cognitive radio networks," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, May 2014, pp. 3933–3937.

[23] 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*, vol. 65, no. 1, pp. 118–129, Jan. 2018.

[24] ——, "Design and implementation of flexible and reconfigurable SDF-based FFT chip architecture with changeable-radix processing elements," *IEEE Trans. Circuits Syst. I*, vol. 65, no. 11, pp. 3942–3955, Nov. 2018.

[25] 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*, vol. 64, no. 6, pp. 1456–1467, Jun. 2017.
[26] J. Dai and H. Yin, "Design and realization of non-radix-2 FFT prime factor processor for 5G broadcasting in release 16," in *Proc. Int. Symp. Comput. Intell. Design*, Dec. 2020, pp. 406–409.
[27] A. Wang, B. Richards, P. Dabbelt, H. Mao, S. Bailey, J. Han, E. Chang, J. Dunn, E. Alon, and B. Nikolić, "A $0.37mm^2$ LTE/wi-fi compatible, memory-based, runtime-reconfigurable $2^n3^m5^k$ FFT accelerator integrated with a RISC-V core in 16nm finFET," in *Proc. IEEE Asian Solid-State Circuits Conf.*, Nov. 2017, pp. 305–308.
[28] K. Xia, B. Wu, X. Zhou, and T. Xiong, "An efficient prime factor memory-based FFT processor for LTE systems," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2016, pp. 1546–1549.
[29] J. Kim, J. Lee, and K. Cho, "A mixed-radix pipeline FFT processor with trivial multiplications for LTE uplink," in *Proc. IEEE Int. Symp. Consumer Electron.*, Sep. 2016, pp. 57–58.
[30] M. Garrido, F. Qureshi, J. Takala, and O. Gustafsson, "Hardware architectures for the fast Fourier transform," in *Handbook of Signal Processing Systems*, 3rd ed., S. S. Bhattacharyya, E. F. Deprettere, R. Leupers, and J. Takala, Eds. Springer, 2019.
[31] M. Garrido, K. Möller, and M. Kumm, "World's fastest FFT architectures: Breaking the barrier of 100 GS/s," *IEEE Trans. Circuits Syst. I*, vol. 66, no. 4, pp. 1507–1516, Apr. 2019.
[32] J. Hazarika, S. Ahamed, and H. Nemade, "Low-complexity, energy-efficient fully parallel split-radix FFT architecture," *Electron. Lett.*, vol. 58, no. 18, pp. 678–680, Jul. 2022.
[33] I. Amat and J. A. López, "Any-radix efficient fully-parallel implementation of the fast Fourier transform on FPGAs," in *Proc. Conf. Design Circuits Integr. Syst.*, Nov. 2023, pp. 67–72.
[34] C. Burrus, "Index mappings for multidimensional formulation of the DFT and convolution," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 25, no. 3, pp. 239–242, Jun. 1977.
[35] G. Everest and T. Ward, *An Introduction to Number Theory*. London: Springer, 2005.
[36] E. Grosswald, *Congruences*. Boston, MA: Birkhäuser Boston, 1984, pp. 35–65.
[37] K.-F. Xia, B. Wu, T. Xiong, and T.-C. Ye, "A memory-based FFT processor design with generalized efficient conflict-free address schemes," *IEEE Trans. VLSI Syst.*, vol. 25, no. 6, pp. 1919–1929, Mar. 2017.
[38] B. Wang, L. Liu, C. Deng, M. Zhu, S. Yin, Z. Zhou, and S. Wei, "Exploration of benes network in cryptographic processors: A random infection countermeasure for block ciphers against fault attacks," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 2, pp. 309–322, Feb. 2017.
[39] R. Thomas, V. DeBrunner, and L. DeBrunner, "A natively real-valued FFT algorithm," in *Proc. Asilomar Conf. Signals Syst. Comput.*, Nov. 2021, pp. 908–912.
[40] V. M. Bautista and M. Garrido, "A hardware-efficient 1200-point FFT architecture that combines the prime factor and Cooley-Tukey algorithms," in *Proc. Conf. Design Circuits Integr. Syst.*, Nov. 2024, pp. 1–6.
[41] P. Paz and M. Garrido, "Efficient implementation of complex multipliers on FPGAs using DSP slices," *J. Signal Process. Syst.*, vol. 95, pp. 543–550, Apr. 2023.
[42] D. Guinart, "Deterministic analysis of the accuracy in FFT hardware architectures," Master's Thesis, Dept. of Electrical Engineering, Linköping University, Jun. 2012.
[43] M. Garrido, V. M. Bautista, A. Portas, and J. Hormigo, "Advanced quantization schemes to increase accuracy, reduce area, and lower power consumption in FFT architectures," *IEEE Trans. Circuits Syst. I*, vol. 72, no. 1, pp. 203–213, Jan. 2025.

Víctor Manuel Bautista was born in Madrid, Spain, in 1998. He received his Bachelor degree of Engineering in Telecommunication Technologies and Services Engineering in July 2021, and his Master in Electronic Systems Engineering (MUISE) in July 2022 from Universidad Politécnica de Madrid (UPM), Spain. Since September 2022, he is working on his Ph.D. at the Department of Electronic Engineering at UPM. His research interests cover 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 Telecommunications Engineering from the 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. From 2012 to 2019 he was an Associate Professor with the Department of Electrical Engineering, Linköping University. In 2019 he moved back to UPM with a Ramón y Cajal Research Fellowship and since 2024 he has been an Associate Professor at the Department of Electronic Engineering at UPM. So far, he has been the author of more than 50 scientific publications. His research interests include optimized hardware design for signal-processing applications, 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 covers high-performance circuits for real-time computation and designs for small area and low power consumption. He appeared in the "World's Top 2% Scientists List" elaborated by Standford University in 2022, 2023 and 2024.