V. M. Bautista and M. Garrido, "A Hardware-Efficient 1200-point FFT Architecture that Combines the Prime Factor and Cooley-Tukey Algorithms," 2024 39th Conference on Design of Circuits and Integrated Systems (DCIS), Catania, Italy, 2024, pp. 1-6, doi: 10.1109/DCIS62603.2024.10769152.

A Hardware-Efficient 1200-point FFT Architecture that Combines the Prime Factor and Cooley-Tukey Algorithms

Víctor Manuel Bautista
Department of Electronic Engineering
Universidad Politécnica de Madrid
Madrid, Spain
victor.bautista@upm.es

Mario Garrido
Department of Electronic Engineering
Universidad Politécnica de Madrid
Madrid, Spain
mario.garrido@upm.es

***Abstract*—In this paper, we present a hardware-efficient 1200-point single-path delay feedback (SDF) fast Fourier transform (FFT) architecture. Contrary to previous FFT architectures for non-power-of-two (NP2) sizes, which usually require a large number of hardware resources, the proposed approach reduces significantly the rotators that are required in the architecture. This is achieved by combining the prime-factor and Cooley-Tukey algorithms. As a result, the proposed architecture has only one non-trivial rotator in between stages of the architecture.**

The effectiveness of this optimization is demonstrated through experimental results, where the proposed approach achieves a significant improvement with respect to previous 1200-point FFTs in terms of hardware resources. Furthermore, the number of resources used in the proposed FFT and in previous optimized 1024-point FFTs is comparable. This fact is highly relevant since NP2 FFTs have traditionally been much less efficient than power-of-two (P2) FFTs. Thus, with this paper we break this old paradigm, making it possible to achieve with NP2 sizes similar efficiency as with P2 ones.

***Index Terms*—NP2, FFT, PFA, SDF, butterfly, FPGA.**

I. Introduction

The fast Fourier transform (FFT) [1] is an efficient algorithm for computing the discrete Fourier transform (DFT). Introduced by Cooley and Tukey in 1965, the FFT significantly reduced the operations of the DFT and opened a research field on FFT algorithms and architectures that has attracted much interest for decades. Nowadays, the FFT is one of the most important signal processing algorithms with applications in many different areas such as communication systems, radio astronomy, space, radar and audio processing.

The FFT transforms a set of N values in the time domain into the frequency domain, being N the size of the FFT. Traditionally, FFT sizes that are powers of two (P2) have been selected. These sizes have the advantage that can be modeled using the Cooley-Tukey algorithm and the vast majority of the literature in the field deals with these sizes.

In modern communication systems, N, may be a non-power-of-two (NP2) number expressed as a product of powers of two, three and five [2]. Efficient solutions for NP2 FFTs rely on algorithms developed during the 70's and 80's [3]–[5]. These algorithms are called prime factor algorithms (PFAs) and are described using advanced mathematics from the field of numbers theory. Compared to the Cooley-Tukey algorithm, the PFA can only be used when the FFT size can be decomposed into co-prime factors, but has the advantage that it removes the complex multiplications, called twiddle factors, that appear between stages in the Cooley-Tukey algorithm.

Due to the high mathematical complexity of the PFA and the difficulty to apply it to the design of FFT architectures, the progress in the area of the design of NP2 FFTs has been slow. Consequently, it has been common for electronic designers to adapt their FFT implementations to a power-of-two size by using zero-padding even when the application demands a NP2 FFT size [6]. This approach avoids the use of NP2 FFTs, but has an additional cost in latency and memory usage compared to using an FFT whose size corresponds to the exact amount of samples that must be processed.

Apart from using P2 FFTs when NP2 sizes are required, another fact related to NP2 FFTs is that current pipeline hardware designs that implement NP2 FFTs still use the Cooley-Tukey algorithm [7]–[9] instead of the PFA. Although PFA is more resource-efficient due to removing the twiddle factors that appear between stages in the Cooley-Tukey algorithm, its higher mathematical complexity has prevented researchers from using it.

In this paper, we break the current paradigm of using only the Cooley-Tukey algorithm for processing data sets of NP2 sizes and present a 1200-point single-path delay feedback (SDF) FFT architecture with a degree of optimization that has not been reached so far. The proposed approach splits the 1200-point FFT algorithm into a 20-point FFT and 60-point FFT. Each of these sub-FFTs is implemented using the PFA, which removes all the complex multiplications in the FFT

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.

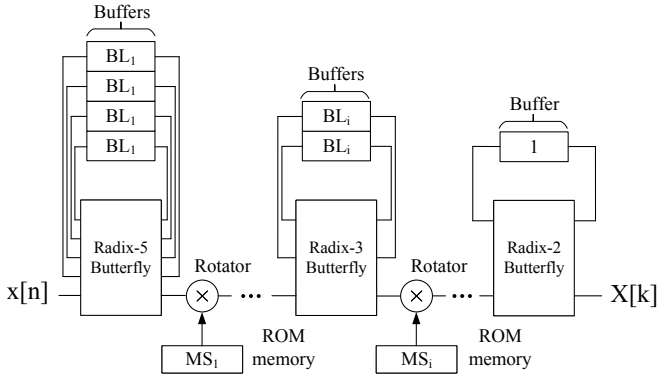

Fig. 1. Example of a generic NP2 SDF FFT architecture.

stages of the sub-FFTs. Then, the sub-FFTs are connected by using the Cooley-Tukey algorithm. As a result, the proposed architecture has only a single stage that requires the calculation of twiddle factors, which reduces significantly the total area of the architecture. This represents a significant improvement compared to previous 1200-point FFT architectures based solely on the Cooley-Tukey algorithm and also with respect to 2048-point FFTs that use zero padding to process a dataset of 1200 samples.

The paper is organized as follows: In Section II, we review the previous knowledge required to understand the proposed approach. In Section III, we describe the proposed architecture. In Section IV, the proposed approach is compared with previous works. In Section V, the experimental results are provided. Finally, in Section VI, the main conclusions are summarized.

II. Background

A. SDF FFT

The DFT transforms an N-point signal from the time domain to the frequency domain according to

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

where $x[n]$ is the signal in the time domain, $X[k]$ is the signal in the frequency domain and W_N^{nk} are the rotations. The Cooley-Tukey FFT algorithm [1] reduces the operations of (1) from $\mathcal{O}(N^2)$ to $\mathcal{O}(N \log N)$. This algorithm divides the original DFT into smaller DFTs, whose sizes are commonly prime numbers. In the case of the FFTs used in 5G, these sizes are two, three and five. These small-sized FFTs are carried out with the so-called butterflies, where a radix-r butterfly calculates an r-point DFT. Apart from the butterflies, rotations are needed to process the FFT. These rotations are essentially complex multiplications.

Fig. 1 shows a generic SDF architecture that processes a non-power-of-two FFT. The architecture is divided into S stages. Each stage i consists of butterflies, buffers and rotators, except for the last stage, which does not require any rotator. The butterflies of radix-r_i calculate an r_i-point FFT. These operations are carried out in parallel using butterflies that have r_i branches. Apart from the butterflies, the SDF architecture

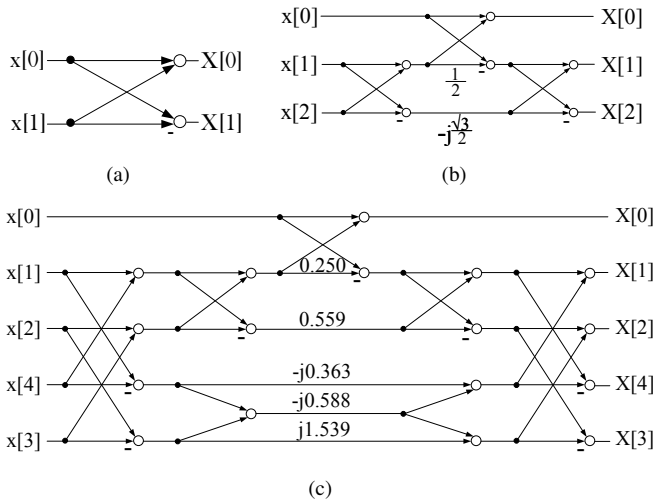

Fig. 2. FFT butterflies. (a) Radix-2. (b) Radix-3. (c) Radix-5.

needs $r_i - 1$ buffers each stage to store the data. The length of the buffers at any stage i is calculated as

$$BL_i = \prod_{j=i+1}^{S} r_j. \quad (2)$$

The rotators calculate complex multiplications by the twiddle factors of the FFT algorithm. These twiddle factors are stored in ROMs of size

$$MS_i = \prod_{j=i}^{S} r_j. \quad (3)$$

Both rotators and buffers have 100% utilization, but butterflies operate a fraction $\frac{1}{r_i}$ of the time [10].

B. FFT Butterflies

A radix-r_i butterfly calculates an r_i-point DFT. Fig. 2(a) shows the flow graph of a radix-2 butterfly. It consists of a complex addition and a complex subtraction. Fig. 2(b) shows the flow graph of a radix-3 butterfly using Rader's algorithm [11]. The algorithm uses 6 complex additions/subtractions and two complex multiplications, where one of them can be implemented with a bit-shift operation in hardware. Finally, Fig. 2(c) shows the radix-5 butterfly using a modified version of the Winograd's algorithm [12] presented before in [10]. It consists of 17 complex additions/subtractions and 5 complex multiplications. The multiplication by 0.25 can be implemented with a bit-shift operation in hardware. Note that the butterflies in Fig. 2 are used at the stages of Fig. 1.

C. Cooley-Tukey Algorithm

Fig. 3(a) shows a 6-point FFT flow graph using the Cooley-Tukey algorithm. The 6-point FFT is divided into two stages. The first stage consists of two radix-3 butterflies and two twiddle factors. The twiddle factors are represented as W_6^1 and W_6^2 and process multiplications by $e^{-j\frac{2\pi}{6}}$ and $e^{-j\frac{4\pi}{6}}$, respectively.

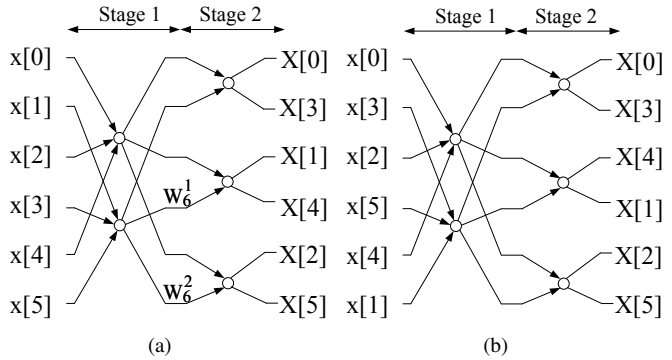

Fig. 3. 6-point FFT flow graphs. (a) Cooley-Tukey algorithm. (b) Prime factor algorithm.

The second stage consists of three radix-2 butterflies. The input order, n, and the output order, k, are calculated as [13]

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

where $n_1, k_1 = \{0, 1, 2\}$, and $n_2, k_2 = \{0, 1\}$ are the residues of r_1 and r_2, respectively. From these residues, the input and output orders are obtained by varying first the less significant bits (LSBs), n_2 and k_2, and then the most significant bits (MSBs), n_1 and k_1.

D. Prime factor algorithm

Fig. 3(b) shows a 6-point FFT flow graph using the prime factor algorithm. This algorithm eliminates the twiddle factors by rearranging the input and output orders. The input and output orders of the 6-point FFT according to the PFA are

$$n \equiv 2 \cdot n_1 + 3 \cdot n_2 \pmod 6, \tag{5a}$$
$$k \equiv 4 \cdot k_1 + 3 \cdot k_2 \pmod 6. \tag{5b}$$

In this particular case, $r_1 = 3$ and $r_2 = 2$ are coprime, so (5) is one of the solutions that solve the algorithm [14].

III. Proposed Approach

A. Overview of the Architecture

Fig. 4 shows the proposed 1200-point SDF FFT architecture. We can identify two radix-5 butterflies, one radix-3 butterfly and four radix-2 butterflies, where each butterfly is connected to a set of buffers. Regarding rotators, we can identify two trivial rotators, represented by $\diamondsuit$, and one general rotator, represented by $\otimes$. The general rotator is connected to a ROM memory that stores 1200 rotation coefficients.

The placement of the butterflies has been decided as $5{\cdot}2{\cdot}2{\cdot}5{\cdot}3{\cdot}2{\cdot}2$ for two reasons. The first reason is that the butterflies are gathered to build a 20-point and a 60-point FFTs. By splitting the 1200-point FFT in these two sub-FFTs, the architecture only requires one non-trivial rotator between them. Internally, both the 20-point and the 60-point FFT only have one trivial rotator each. Thus, the principle that we follow to split the FFT is to place butterflies with the same radix in different sub-FFTs. This way, the factors in the sub-FFTs are coprime and the PFA can be used to remove rotators. The only exception is that two radix-2 blocks are permitted in each sub-FFT, as they only introduce a trivial rotation, which is hardware-efficient.

The second reason for the butterfly order is that, in the sub-FFTs, the radices are ordered by placing bigger radices first. Thus, radix-5 butterflies are placed first, then radix-3 butterflies and radix-2 butterflies at the end. This way, the last stages of the architecture, which have short buffers, use only radix-2. This places radix-5 and radix-3 butterflies in stages with longer buffers, which can be moved inside the butterflies to pipeline them.

B. Algorithm Selection

The 1200-point FFT combines the prime factor and the Cooley-Tukey algorithms. The 20-point and 60-point sub-FFTs are designed using the prime factor algorithm in [14]. In the factorizations $20 = 5{\cdot}2{\cdot}2$ and $60 = 5{\cdot}3{\cdot}2{\cdot}2$, only the factor 2 is repeated. Thus, we can construct an algorithm using the PFA that considers only trivial rotations and eliminates non-trivial rotations.

Fig. 5 shows the flow graph of the 20-point FFT using the PFA algorithm proposed in [14]. The index mapping used in this flow graph is

$$n_{20} \equiv 4 \cdot n_1 + 10 \cdot n_2 + 5 \cdot n_3 \pmod{20}, \tag{6a}$$
$$k_{20} \equiv 16 \cdot k_1 + 5 \cdot k_2 + 10 \cdot k_3 \pmod{20}. \tag{6b}$$

The coefficients of (6) have been carefully selected to eliminate the twiddle factors that appear between the radix-5 stage and the first radix-2 stage. The twiddle factors that appear between the consecutive radix-2 stages have the value $\phi = 5$, which correspond to a rotation by [15]

$$e^{-j\frac{2\pi}{N}\phi} = e^{-j\frac{2\pi}{20}5} = e^{-j\frac{\pi}{2}} = -j. \tag{7}$$

This is a trivial rotation that can be calculated by exchanging the real part and imaginary parts of the input and then changing the sign of the resulting imaginary part.

Fig. 6 shows the flow graph of the 60-point FFT using the PFA algorithm in [14]. The index mapping used in this flow graph is

$$n_{60} \equiv 36 \cdot n_4 + 40 \cdot n_5 + 30 \cdot n_6 + 15 \cdot n_7 \pmod{60}, \tag{8a}$$
$$k_{60} \equiv 12 \cdot k_4 + 20 \cdot k_5 + 45 \cdot k_6 + 30 \cdot k_7 \pmod{60}. \tag{8b}$$

Analogously to the proposed 20-point FFT, the coefficients have been carefully selected to eliminate the twiddle factors of the first and second stages. The twiddle factors of the last stage correspond to $\phi = 15$, which results in $e^{-j\frac{2\pi}{60}15} = -j$. This means that the algorithm uses only trivial rotations.

C. Connecting the Radix-20 and Radix-60 FFTs

The 1200-point FFT is built by connecting the 20-point and 60-point sub-FFTs. As the factorization of 20 and 60 share common factors, it is not possible to apply the PFA and, therefore, we have to use the Cooley-Tukey algorithm. The

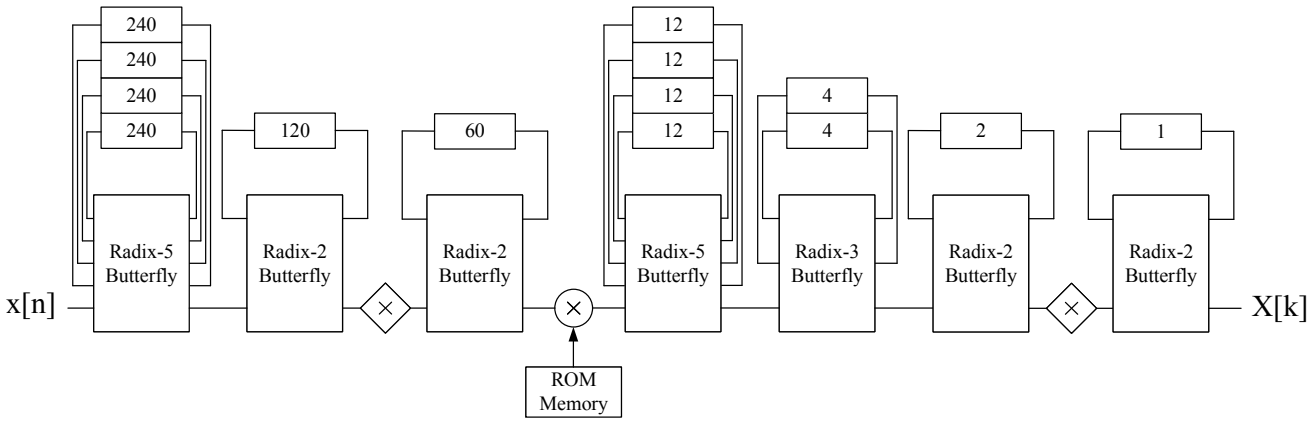

Fig. 4. Proposed 1200-point SDF FFT architecture using the proposed algorithm.

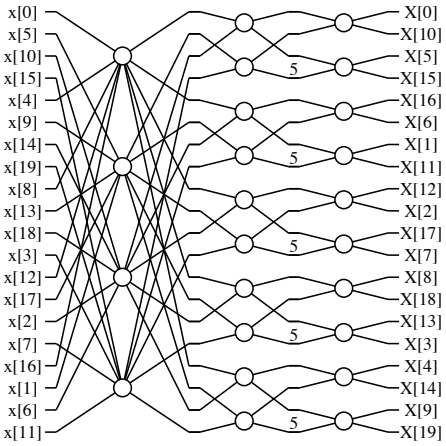

Fig. 5. Flow graph of the proposed 20-point FFT using PFA.

digit representation of the inputs and outputs of the 1200-point FFT algorithm is $n_1 n_2 ... n_7$ and $k_1 k_2 ... k_7$, respectively. Each digit operates on the stage $i = 1, 2, ..., 7$ and has the values $0, 1, ..., r_i - 1$. The index mapping of the 1200-point FFT is

$$n_{1200} \equiv 60 \cdot n_{20} + n_{60} \pmod{1200}, \tag{9a}$$

$$k_{1200} \equiv k_{20} + 20 \cdot k_{60} \pmod{1200}. \tag{9b}$$

Note that this index mapping is analogous to the simple 6-point FFT of example in (4) differing in the radices. The first three digits, n_1, n_2, n_3 and k_1, k_2, k_3, operate on the radix-20 butterfly according to (6) and the last four digits, n_4, n_5, n_6, n_7 and k_4, k_5, k_6, k_7, operate on the radix-60 butterfly according to (8). This makes n_{20} and n_{60} be extracted according to the PFA and also be evaluated modulo 20 and 60, respectively.

D. Number of Components

Table I shows the number of real adders, memory, and real multipliers of each component of the proposed architecture. The constant multiplications of the radix-3 and radix-5 butterflies are implemented with shift-and-add operations. This increases the number of real adders from 12 to 18 in the radix-3 butterfly, and from 34 to 54 in the radix-5 butterflies. Despite the increase of the real adders, it is worth noting

TABLE I
NUMBER OF ELEMENTS OF THE PROPOSED ARCHITECTURE.

	Units	Real adders	Real mults.	Memory size
Radix-2 butterfly	4	4	0	0
Radix-3 butterfly	1	18	0	0
Radix-5 butterfly	2	54	0	0
General rotator	1	3	3	0
Trivial rotator	2	1	0	0
Buffers	-	0	0	1199
ROM memory	1	0	0	1200
Total sum	-	147	3	2399

TABLE II
COMPARISON OF THE PROPOSED ARCHITECTURE WITH PREVIOUS ONES.

	[6]	[8]	Prop.
Arch. type	MP*	SDF	SDF
N	2048	1200	1200
Real adders	93	114	147
Real mults.	24	36	3
Mem. size (buffers)	2136	1199	1199
Mem. size (ROMs)	510	1516	1200
Mem. size (Total)	2646	2715	2399
Latency (cycles)	1200	1255	1234

that no real multipliers are implemented and, then, no DSP slices are used in the butterflies. The general rotators' design is based on [16]. It uses three real adders and 3 real multipliers. Both the adders and multipliers are encapsulated into three DSP slices. The trivial rotators use one real adder. Regarding storage, the 1200-point FFT has buffers with a total length of 1199 complex words and the single ROM memory stores 1200 rotation coefficients, leading to a total storage of 2399 complex words. The total sum at the bottom of Table I is obtained by multiplying the number of components by the units that use these components and then adding the result.

IV. COMPARISON

Table II compares the proposed approach with recent designs. The approach in [6] is a 2048-point modified pipeline (MP) FFT architecture similar to an SDF FFT. Although $N = 2048$ in [6], it processes only 1200 data and the rest

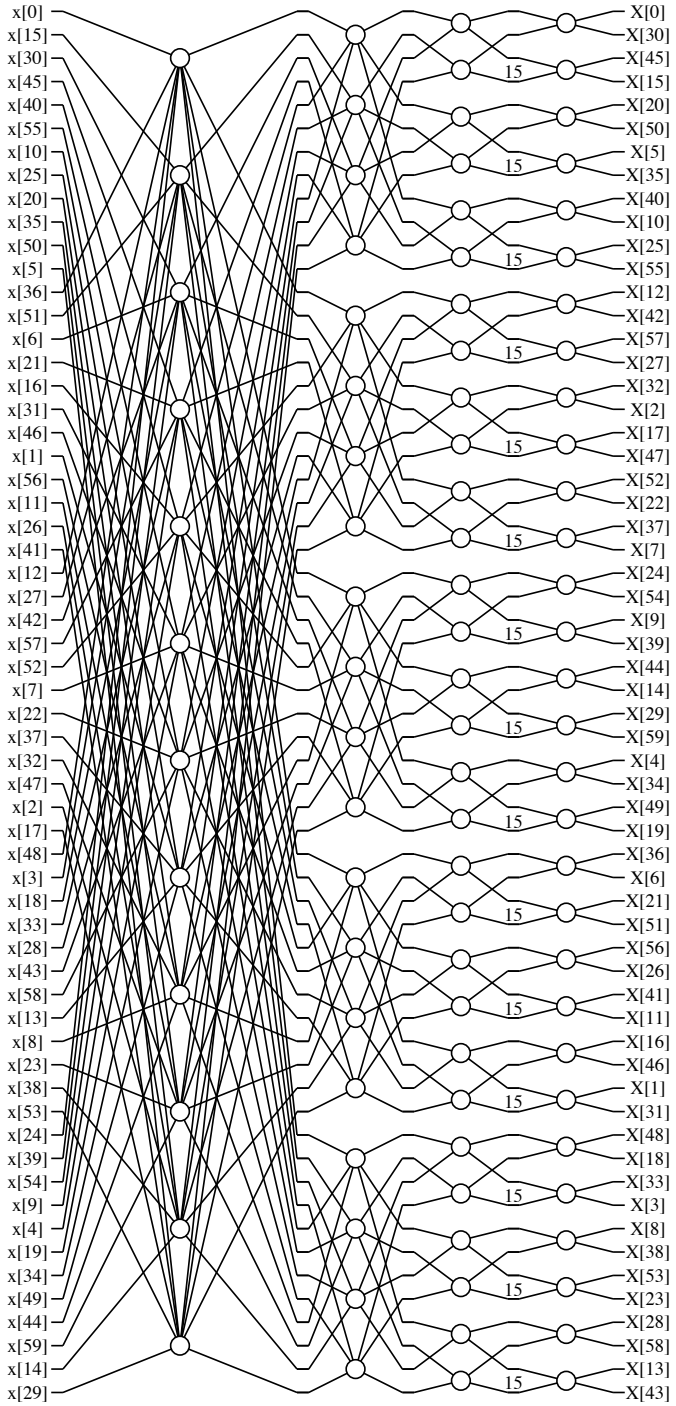

Fig. 6. Flow graph of the proposed 60-point FFT using PFA.

of the inputs are set to zero representing the OFDM guard bands. This makes it comparable to the proposed approach, as both process the same amount of inputs. The other design to compare with is [8]. This design is a 1200-point SDF FFT architecture that is based on the Cooley-Tukey algorithm.

The figures of merit in Table II are real adders, real multipliers, the size measured in complex words of the buffers and the size in complex words of the ROM memories. The amount of real adders and real multipliers used in the rotators of [6] has been extracted from [17]. The trivial rotator used in [6] counts as one real adder as in the proposed design.

Regarding real adders, the 1200-point FFTs include more logic in the radix-3 and radix-5 butterflies. Therefore, [8] and the proposed architecture require more real adders than the 2048-point FFT in [6]. The proposed architecture has 33 more real adders than [8] because the constant multiplications in the butterflies are implemented with shift-and-add operations, whereas in [8] they use full real multipliers.

Regarding real multipliers, [6] uses three real multipliers per rotator, with 24 real multipliers in total since only the first 8 stages use generic rotators. [8] uses three real multipliers per rotator, 8 for radix-5 butterflies, and 2 for radix-3 butterflies, resulting in 36 real multipliers. The proposed architecture reduces this number to only three real multipliers by using PFA and implementing butterfly constant multiplications with shift-and-add operations.

In [6], the memory size of the buffer is 2136 complex words, consisting of 2047 words from the buffers of a typical 2048-point SDF FFT architecture plus an extra 89 words to avoid memory conflicts. Both [8] and the proposed designs use 1199 words for buffer memory, typical for a 1200-point SDF FFT architecture. Regarding ROM size, [6] uses general rotators in the first 8 stages, with a total ROM size of 510 words, which is smaller than in [8] and the proposed architecture, but requires additional logic to divide the coefficients into 8 sets. [8] stores the conventional coefficients of a 1200-point FFT SDF architecture, while the proposed architecture reduces this to one memory of 1200 positions, equivalent to just the first ROM in [8]. Despite the ROM size optimizations in [6], the proposed architecture achieves the lowest total memory size considering the sum of both memory contributions.

Regarding latency, the proposed architecture presents a slight increase of 34 cycles compared to the architecture of [8]. This comes from applying pipeline to the design in order to achieve high clock frequencies in the implementation. Compared to [8], the proposed architecture presents a reduction of 11 clock cycles.

As a result, the removal of general rotators in the proposed architecture reduces significantly the number of real multipliers and ROM memories.

V. Experimental Results

In order to show the capabilities of the proposed design, it has been implemented on a Virtex Ultrascale+ FPGA. The chip name is XCVU37P-FSVH2892-2L-E. Table III compares the experimental results of the proposed approach with respect to previous works. The figures of merit included in the table are word length (WL), area, clock frequency (f_{CLK}), signal-to-quantization-noise ratio (SQNR), latency and power consumption. The table provides the power consumption in mW, and the power normalized by the FFT size and the clock frequency (N. Power) as $\frac{\text{mW}}{N \cdot \text{MHz}}$.

The works [8], [18], [19] present 1024-point SDF FFT architectures. The proposed architecture reduces the number of DSP slices with respect to them by a factor 4 or larger

TABLE III
EXPERIMENTAL RESULTS OF SDF FFT ARCHITECTURES.

	[18]	[17]	[19]	[8]	[8]	Prop.
Architecture Type	SDF	MP	SDF	SDF	SDF	SDF
N	1024	2048	1024	1024	1200	1200
FPGA	S7	V7	V7	VU+	VU+	VU+
WL	16	12	16	16	16	16
LUT	2427	3668	1849	2305	3050	4207
FF	3045	3773	1292	1821	3637	4479
CARRY8	-	-	-	120	288	442
CLB	-	-	-	533	690	877
DSP	12	34	12	27	36	3
BRAM	4	11	4	2	5	4
f_{CLK} (MHz)	100	-	360	500	500	500
SQNR (dB)	-	-	56.25	54.43	43.27	46.25
Latency (cycles)	-	1200	1063	1087	1255	1234
Latency (μs)	-	-	2.95	2.17	2.51	2.47
Power (mW)	-	-	312	490	651	524
N. Power $\left(\frac{\text{mW}}{N\cdot\text{MHz}}\right)$	-	-	0.846	0.957	1.085	0.873

due to using only one general rotator, instead of at least four general rotators in [8], [18], [19]. This is compensated by using more LUTs than [8], [18], [19]. As a result, the proposed 1200-point architecture is comparable in area to 1024-point ones [8], [18], [19]. This is a great conclusion, as we have been able to develop an NP2 FFT that is as resource-efficient as P2 ones. Additionally, the latency in [8], [18], [19] and the proposed approach is close to N and, thus, equivalent. The SQNR of the proposed 1200-point FFT is high but lower to that in [8], [19]. Finally, the normalized power consumption of the proposed architecture is slightly higher than that in [19] and a 9% lower than than that in [8].

The work [17] presents the experimental results of a 2048-point MP FFT architecture detailed in [6]. Reducing the general rotators in the proposed design causes a 91% reduction of the DSPs compared to [17]. The reduction of memory of the proposed architecture compared to [17] is translated into 63% less BRAMs in the proposed design. Therefore, the proposed 1200-point FFT is significantly more efficient than the architecture in [17] for processing data sets of 1200 point.

Finally, compared to the 1200-point FFT in [8], the proposed 1200-point FFT improves most figures of merit while maintaining the same clock frequency. The number of DSPs is drastically reduced from 36 to 3 at the cost of a less significant increase in LUTs and FFTs. The SQNR increases 3 dB, the latency is reduced slightly, and the power consumption is reduced by 19%. All these improvements are the result of removing a large number of rotators.

In conclusion, experimental results show that the proposed 1200-point FFT improves area and memory compared to previous designs that process 1200 data [6], [8]. Furthermore, whereas P2 FFTs have been traditionally more efficient than NP2 ones, the proposed NP2 FFT achieves similar efficiency than P2 ones, opening the new research line of making NP2 FFTs as efficient as P2 ones.

VI. CONCLUSIONS

This work has presented a hardware-efficient 1200-point SDF FFT architecture. The algorithm combines the PFA and the Cooley-Tukey algorithm. This allows for removing most of the rotators that appear between the FFT stages in P2 FFTs and other previous NP2 FFTs. As a result, the proposed architecture is highly efficient in terms of hardware resources. This represents a significant improvement with respect to previous NP2 FFT, which were not hardware-efficient so far. In fact, previous NP2 FFT were traditionally much less efficient than P2 ones. However, the proposed approach demonstrates that it is feasible to achieve NP2 FFTs with similar efficiency in terms of resources and performance than P2 ones.

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

[4] 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.

[5] 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.

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

[7] Y. Zhou, S. Li, C. Yang, J. Kuang, J. Chen, and C. Zhang, "Hardware efficient radix-3/4/11 based FFT processor for 5G NR application," in *Proc. Asilomar Conf. Signals Syst. Comput.*, Oct. 2022, pp. 634–639.

[8] 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 Integrated Syst.*, Nov. 2023, pp. 1–6.

[9] 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*, vol. 65, no. 11, pp. 3942–3955, Nov. 2018.

[10] 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.

[11] 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.

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

[13] S. K. Mitra, *Discrete Signal Processing: A Computer-Based Approach*, 4th ed. McGraw-Hill, 2011.

[14] "A method to obtain non-power-of-two FFT flow graphs based on a new prime factor algorithm," *Under Review*.

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

[16] 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.

[17] M. Mahdavi, O. Edfors, V. Öwall, and L. Liu, "A low latency and area efficient FFT processor for massive MIMO systems," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2017, pp. 1–4.

[18] V. M. Milovanović and M. L. Petrović, "A highly parametrizable chisel HCL generator of single-path delay feedback FFT processors," in *Proc. IEEE Int. Conf. Microelectron.*, Sep. 2019, pp. 247–250.

[19] 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*, 2024, Early Access.