Advanced Quantization Schemes to Increase Accuracy, Reduce Area, and Lower Power Consumption in FFT Architectures

Mario Garrido, *Senior Member, IEEE*, Víctor Manuel Bautista, Alejandro Portas, and Javier Hormigo

***Abstract*— This paper explores new advanced quantization schemes for fast Fourier transform (FFT) architectures. In previous works, FFT quantization has been treated theoretically or with the sole aim of improving accuracy. In this work, we go one step beyond by considering also the implications that quantization schemes have on the area and power consumption of the architecture. To achieve this, we have analyzed the mathematical operations carried out in FFT architectures and explored the changes that benefit all the figures of merit. By combining or alternating truncation and rounding, and using the half-unit biased (HUB) representation in the different computations of the architecture, we have achieved quantization schemes that increase accuracy, reduce area, and lower power consumption simultaneously. This win-win result improves multiple figures of merit without worsening any other, making it a valuable strategy to optimize FFT architectures.**

***Index Terms*— Fast Fourier transform (FFT), half-unit biased (HUB) representation, accuracy, single-delay feedback (SDF).**

I. Introduction

THE fast Fourier transform (FFT) is one of the most crucial signal-processing algorithms. It is used in a wide range of applications in areas such as communication systems [1], [2], [3], [4], radio astronomy [5], [6], [7], and medical imaging [8], [9], [10].

Over the last 20 years, numerous hardware FFT architectures have been proposed. The main design goals have been to reduce the area of the FFT and increase the throughput. On the one hand, the area has been reduced by presenting new architectures with more efficient use of the hardware resources [11], by implementing the rotators as shift-and-add operations [12], and by allocating these rotators in such a way that their number and complexity are reduced [13], [14]. On the other hand, the throughput has been increased thanks to the use of parallel [1], [14], [15] FFT architectures.

Despite the large number of works on low-area or high-throughput FFT architectures, not so many works in the literature optimize FFT architectures based on the accuracy of the computations. Among them, some works deal with the scaling of the data at the stages of the FFT architecture [16], [17]. These works allow for a different word length at each stage of the FFT computation. This makes it possible to choose the word length profiles that lead to the highest accuracy with the smallest use of resources. Other works analyze the accuracy in real-valued FFTs [18], [19], instead of complex-valued ones. In other approaches, the accuracy is improved by scaling the rotation coefficients [20], [21]. In these works, exploring multiple alternatives for the rotation coefficients leads to more accurate rotations in the FFT. Finally, some works modify the quantization scheme concerning the conventional truncation [22], [23], [24]. These works provide solutions that compensate for the quantization bias by alternating truncation and rounding, which leads to higher accuracy.

In this work, we analyze new quantization schemes based on rounding and truncation and incorporate the half-unit biased (HUB) representation system in the computations. To derive these new quantization schemes we have looked not only at the accuracy improvement but also at the impact on the hardware resources of the architecture. Contrary to previous approaches, which improve accuracy at the cost of increasing area and power consumption, we have derived quantization schemes that increase the accuracy of the computations and simultaneously reduce area and power consumption. To achieve this, we classified the components of the FFT into several groups: even, odd, first and last butterflies; and general and trivial rotators. Then, we applied different truncation and rounding schemes to them and evaluated their impact on accuracy, hardware resources, and power consumption.

Furthermore, we have considered using the half-unit biased (HUB) representation system. The HUB format is based on assuming that a logic '1' is appended to the binary numbers that represent the data. This '1' leads to numbers with one extra bit. However, this extra bit is not represented in a physical bit of information. Additionally, in fixed-point designs, the HUB approach allows for calculating round-to-nearest with no

Manuscript received 4 March 2024; revised 26 April 2024 and 27 May 2024; accepted 27 June 2024. This work was supported in part by MCIN/AEI/10.13039/501100011033 and "ERDF A Way of Making Europe" under Project PID2021-126991NA-I00, in part by European Union Next Generation EU/PRTR under Project TED2021-131527B-I00, in part by the Fondo Europeo de Desarrollo Regional under Grant UMA20-FEDERJA-059, and in part by MCIN/AEI/10.13039/501100011033 and "ESF Investing in Your Future" under Grant RYC2018-025384-I. This article was recommended by Associate Editor L. Gan. *(Corresponding author: Mario Garrido.)*

Mario Garrido, Víctor Manuel Bautista, and Alejandro Portas are with the Department of Electronic Engineering, ETSI de Telecomunicación, Universidad Politécnica de Madrid, 28040 Madrid, Spain (e-mail: mario.garrido@upm.es; victor.bautista@upm.es; alejandro.portas.fernandez@alumnos.upm.es).

Javier Hormigo is with the Department of Computer Architecture, Universidad de Málaga, 29071 Málaga, Spain (e-mail: fjhormigo@uma.es).

Digital Object Identifier 10.1109/TCSI.2024.3421348

additional hardware cost compared to conventional truncation. This either results in an improvement in accuracy or allows for reducing the word length and, therefore, the area and delay of the circuit [25], [26]. In floating-point designs, this simplification improves the implementation of arithmetic units directly [27], [28], [29]. Additionally, the HUB format has been successfully used to improve the accuracy and reduce the complexity of other signal processing algorithms implemented in hardware, such as the QR decomposition [25]. However, the HUB format has not been applied to the FFT. This work is the first in this line and the first to apply HUB to complex numbers.

To apply the different quantization schemes to the FFT, we have considered the single-path delay feedback (SDF) FFT [11], one of the most widely used FFT architectures. The SDF is a pipelined FFT architecture that processes one sample per clock cycle in a continuous flow. This provides a good trade-off between throughput and area. Despite considering the SDF FFT architecture for the analysis in the paper, it is worth realizing that the accuracy of the FFT is independent of the architecture that we use: As long as the mathematical calculations, i.e., additions and multiplications, are carried out using the same quantization scheme, in terms of accuracy it does not matter if these computations are carried out in series, as in the SDF FFT [30], in parallel, as in multi-path delay commutator (MDC) [31], [32], multi-path serial commutator (MSC) [33], and multi-path delay feedback (MDF) [34] FFTs, or iteratively, as in memory-based (MB) FFTs [35]. For all of them, the same quantization scheme leads to the same value at each output frequency.

Experimental results for a 16-bit radix-2^2 1024-point SDF FFT have been carried out to analyze different quantization schemes. It has been observed that a careful selection of the quantization scheme results in implementations that achieve higher accuracy, lower area, and lower power consumption simultaneously. This represents a win-win situation regarding accuracy, area, and power consumption. Consequently, exploring quantization schemes becomes a key factor for optimizing FFT architectures.

Another advantage of the proposed approach concerning previous theoretical works is that our work is based on actual experimental results. This provides exact results from the architectures and allows for including other figures of merit in the analysis, such as area and power consumption, which is impossible in a theoretical quantization analysis. This way, this paper offers a global study that integrates all the figures of merit of all analyzed architectures.

This paper is organized as follows. Section II reviews the HUB format and the SDF FFT architecture. In Section III, we show how rounding, truncation, and the HUB format are applied to the different operations in the FFT. Section IV describes the quantization schemes we analyzed in this paper. Section VI provides the experimental results for the different configurations and analyzes them. In Section VII, we analyze the influence of other FFT parameters in the SQNR. In Section VIII, we compared the proposed architectures with previous works. Finally, in Section IX, we summarize the paper's main conclusions.

II. Background

A. The HUB Representation System

The HUB representation system is a new family of formats that allow for optimizing computations with real numbers by simplifying rounding to nearest and two's complement operations [36]. It is based on shifting the values exactly represented under conventional formats by half of the weight of the least significant bit (LSB). In practice, HUB numbers are like conventional ones but append an implicit least significant bit (ILSB) to the binary number to get the represented value. This ILSB is constant and equal to one [36]. For example, the HUB number 1.1010 represents the value 1.10101. This hidden LSB (the ILSB) must not be stored or transmitted. It only has to be considered when an operation with that number is carried out. Thus, the HUB format has the same number of explicit bits and the same accuracy as the conventional one.

The main advantage of using HUB numbers is that rounding to the nearest is performed simply by truncation. For example, the nearest 4-bit HUB number to the value 1.0101101 is 1.010 (which represents 1.0101), whereas, for a conventional representation, it would be 1.011. Generally, the error in a particular example is different for traditional and HUB formats. However, the bounds of the quantization errors for both approaches are the same [36]. Therefore, although HUB and conventional approaches provide different values, both representations allow for the same accuracy.

Another essential advantage of the ILSB is that the two's complement of a HUB number is implemented simply by inverting all explicit bits (one's complement) [36]. For conventional fixed-point numbers, the two's complement operation requires a bit-wise inversion plus the addition of one unit-in-the-last-place (ULP). Conversely, in the HUB approach, the ILSB absorbs the effect of the required increment, and no addition needs to be calculated. This significantly reduces the logic required to implement the two's complement of a HUB number.

Finally, the conversion between conventional and HUB formats is almost trivial. A HUB number is converted to a conventional one simply by explicitly appending the ILSB. Note that HUB is a store/transmission format, meaning that a HUB number must be converted to a conventional format (explicitly or virtually) before operating with it. Consequently, HUB operators generally start by appending the ILSB to the input values, which transforms them into conventional values that are operated regularly. Conversely, a conventional number could be transformed into a HUB number with less bit-with by simply truncating it. Therefore, HUB and conventional numbers can be quickly and effectively combined in the same design as seen in the following sections. Note, however, that the conversion between a conventional and a HUB number with the same bit-width always causes a loss of accuracy. Therefore, converting a conventional number to a HUB one should be performed when its bit-width has to be reduced. This typically occurs after performing an arithmetic operation that produces a bit-width growth, such as addition or multiplication.

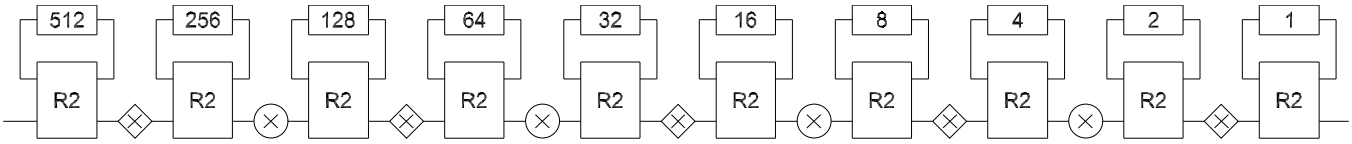


Fig. 1. Radix-2^2 1024-point SDF FFT architecture.

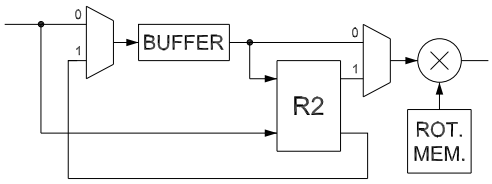


Fig. 2. Internal structure of a stage in an SDF FFT architecture.

B. FFT Architecture Under Study

Fig. 1 shows the FFT architecture under study. It is a 1024-point radix-2^2 SDF architecture [11]. It consists of $n = \log_2(N) = \log_2(1024) = 10$ stages, where N is the FFT size. Each stage, $s \in \{1, \ldots, n\}$, includes a butterfly (R2), a buffer of length 2^{n-s}, and a rotator. In the architecture, there are two types of rotators: trivial and general. Trivial rotators are diamond-shaped and calculate rotations by the angles $0°$ and $-90°$. These rotations are called trivial because they are computed by simply exchanging the real and imaginary parts of the data and/or changing their signs. General rotators are represented with the symbol ($\otimes$) and calculate rotations by multiple angles. The implementation of general rotators often involves complex multipliers.

The internal structure of a stage in the SDF FFT is shown in Fig. 2. First, the buffer collects 2^{n-s} data. Then, these data are added and subtracted with the incoming data in the butterfly. The results of the additions pass to the rotator, while the results of the subtractions are fed back to the buffer. When these data leave the buffer, they are sent to the rotator. This occurs at the same time that data for a new FFT calculation arrive at the input of the stage, which allows for continuous flow processing.

III. Quantization in the FFT

This work considers truncation, rounding, and the HUB format for the quantization schemes under study. This section describes how to adapt the FFT operations accordingly.

We assume that the FFT architecture is embedded in a conventional number format system to have a general framework for the study. Therefore, we consider conventional input and output signals in all the configurations, and we only work with HUB numbers internally in those quantization schemes that use HUB. The reason for taking this approach is to compare all the quantization schemes under the same circumstances.

Note also that the only difference between a HUB number and a conventional one is the existence or absence of the ILSB, which is not physically present. Consequently, some of the modifications described in this section are simply conceptual and do not require any actual modification of the specific logic circuit.

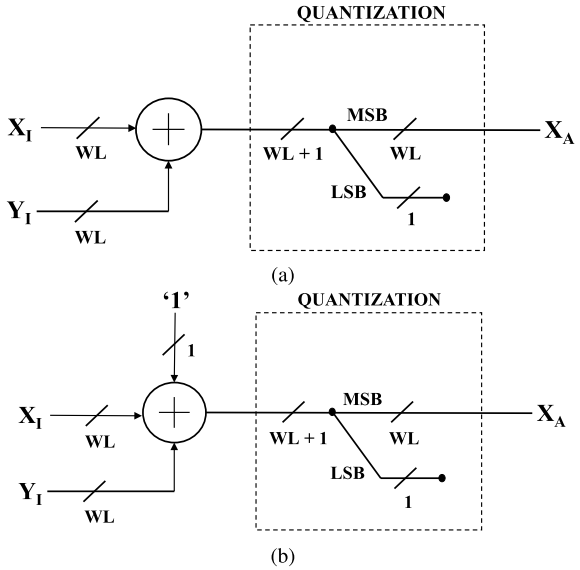


Fig. 3. Mathematical operations in the adders of the FFT butterflies. (a) Adder in conventional fixed-point representation using truncation. (b) Adder in both HUB and conventional fixed-point representation using rounding.

This section analyzes how the different parts of the FFT architecture are treated depending on the format. This includes adapting the inputs, butterflies, general rotators, trivial rotators, and outputs.

A. Adaptation of the Input From Conventional to HUB Format

When HUB format is used in the FFT, the conventional input signal is turned into HUB format in the first operator of the FFT unit, which is a butterfly. This first butterfly is particular because it has conventional input and HUB output. No actual physical (or logical) change is needed to achieve this. The butterfly circuit is the same as the conventional one explained next in Section III-B. However, its output values are considered as HUB numbers, with an ILSB set to one that must be considered in the next arithmetic unit.

B. Adaptation of the Butterflies

1) Truncation: The implementation of conventional butterflies requires one addition and one subtraction for the real parts and another addition and subtraction for the imaginary parts. When adding or subtracting two numbers with WL bits, the output of the adder requires $WL + 1$ bits to represent all the possible output values. To keep the word length of the data constant throughout the FFT, the LSB after the adders in the butterflies is truncated. This corresponds to the equation

$$X_A = \left\lfloor \frac{X_I \pm Y_I}{2} \right\rfloor, \tag{1}$$

where X_I and Y_I are the inputs, X_A is its output, $(\pm)$ is a plus in the case of an adder and a minus in the case of a subtractor, and $(\lfloor\cdot\rfloor)$ is the floor operation. Fig. 3(a) shows the case of adding two numbers. Note that we consider all the binary numbers as integer numbers, following the convention in [20]. According to it, any binary number can be considered an integer number scaled by a power of two, so we can treat the binary number as an integer and apply the scaling afterward.

2) Rounding: The mathematical operation in case of rounding is

$$X_A = \left\lceil \frac{X_I \pm Y_I}{2} \right\rceil = \left\lfloor \frac{X_I \pm Y_I + 1}{2} \right\rfloor, \tag{2}$$

where $(\lceil\cdot\rceil)$ is the ceil operation. Again, $(\pm)$ is a plus in the case of an adder and a minus in the case of a subtractor. In the case of the adder, the circuit that calculates this operation is shown in Fig. 3(b). The equivalence between rounding up and adding one plus truncating in (2) is possible because only one bit is removed.

3) HUB: When the inputs are in HUB format, the input numbers have an implicit 1, which corresponds to adding 0.5 to both X_I and Y_I. Thus, the output of an adder is obtained as

$$X_A = \left\lfloor \frac{X_I + 0.5 + Y_I + 0.5}{2} \right\rfloor. \tag{3}$$

Mathematically, (3) is the same equation as (2) for the case of addition, so the circuit for rounding and HUB is the same in the case of addition, which is shown in Fig. 3(b). However, note that in the case of rounding, the circuit is obtained from rounding the data, whereas for HUB, the circuit is the result of using HUB inputs.

Contrary to the HUB addition, the implicit bits cancel each other in the HUB subtraction. This results in

$$X_A = \left\lfloor \frac{X_I - Y_I}{2} \right\rfloor, \tag{4}$$

so the truncation and HUB format calculations in the subtractors are the same.

As a final remark, regardless of whether the inputs are conventional or HUB numbers, to produce a HUB output, the outputs of both adder and subtractor are truncated as in the conventional output circuit to keep the word length while avoiding overflow. No modification is required at the output to get HUB output values. In this case, the output values are simply considered HUB numbers with an ILSB. However, in this case, the truncation carries out an actual rounding-half-up thanks to the ILSB.

C. Adaptation of the General Rotators

1) Truncation: In a conventional fixed-point representation using truncation, general rotators calculate

$$X_O = \left\lfloor \frac{X_B \cdot (C + jS)}{R} \right\rfloor, \tag{5}$$

where $C + jS$ is the rotation coefficient, R is the magnitude or scaling of this coefficient, and $X_B = X_{Br} + jX_{Bi}$ is the complex output of the butterfly that must be rotated. The

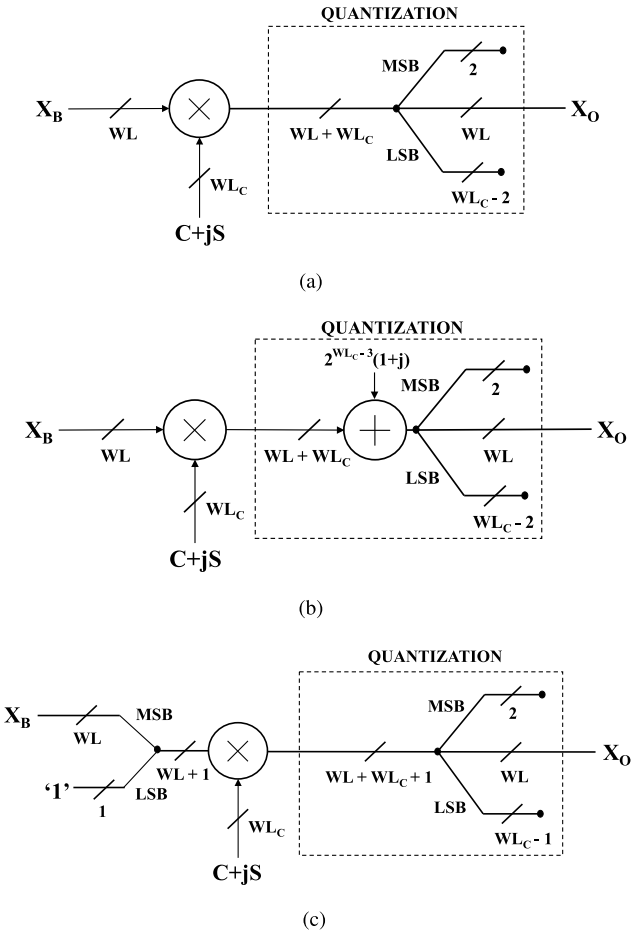


Fig. 4. FFT rotators using multipliers. (a) Conventional fixed-point representation. (b) Rounding. (c) HUB format.

circuit calculating the rotation is shown in Fig. 4(a). The coefficients are represented with WL_C bits and their scaling is $R = 2^{WL_C-2}$. This scaling is applied to guarantee that the coefficients can be represented with WL_C bits. Note that WL_C bits have a range of values $[-2^{WL_C-1}, 2^{WL_C-1} - 1]$. Therefore, if the scaling of the rotator were $R = 2^{WL_C-1}$, it would not be possible to represent the value 2^{WL_C-1} with WL_C bits.

As an FFT rotator calculates a rotation in the complex plane, it only modifies the input signal's phase and preserves its magnitude. Therefore, as the rotation coefficients scale the signal by $R = 2^{WL_C-2}$, this scaling must be compensated to set the output signal with the same magnitude as the input. This is why the $WL_C - 2$ LSBs are removed from the output of the rotator in Fig. 4(a). The middle WL bits are the output of the rotator, and the 2 MSBs are removed. Note that the output signal never reaches the level of these 2 MSBs since the magnitude of the coefficients is $R = 2^{WL_C-2}$.

2) Rounding: Rounding in the rotators is calculated as

$$X_O = \left\lceil \frac{X_B \cdot (C + jS)}{R} \right\rceil, \tag{6}$$

which is equivalent to

$$X_O = \left\lfloor \frac{X_B \cdot (C + jS) + \frac{R}{2}(1 + j)}{R} \right\rfloor. \tag{7}$$

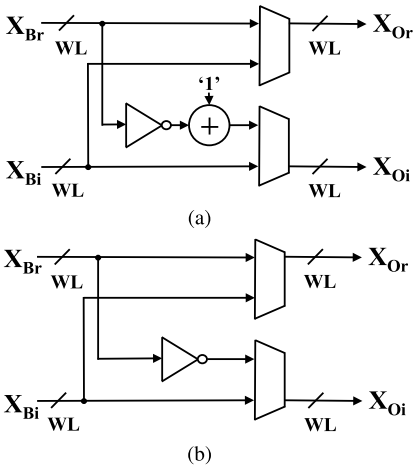


Fig. 5. Trivial rotators in the FFT. (a) Conventional fixed-point representation. (b) HUB format.

The corresponding circuit is shown in Fig. 4(b). Note that $R/2 = 2^{WL_C-3}$. By adding $\frac{R}{2}(1+j)$ after the multiplication, we are adding one unit to the bit that is placed to the right of the bit that will end up as LSB after removing the least significant $WL_C - 2$ bits. This creates the effect of rounding. Note that, as numbers are complex, $R/2$ is added to their real and imaginary parts through the factor $(1+j)$.

3) HUB: If the input signal of the rotator is a HUB number, the circuit that calculates the complex multiplication is shown in Fig. 4(c) and its mathematical operation is

$$X_O = \left\lfloor \frac{(X_B + 0.5 + j0.5) \cdot (C + jS)}{R} \right\rfloor. \quad (8)$$

As the input X_B is in HUB format and the coefficient is a conventional fixed-point number, the implicit bit is included in the operation to carry out the rotation. This corresponds to adding $(0.5 + j0.5)$ to X_B, which is done in hardware by appending a '1' to the LSBs of the real and imaginary parts of X_B.

Similarly to the butterfly case, regardless of the input format, the HUB outputs for the rotators are obtained by truncating the outputs of the multipliers. This truncation produces a rounded-half-up HUB number.

D. Adaptation of the Trivial Rotators

1) Conventional Representation: Trivial rotators calculate rotations by 0° and −90°. The hardware implementation of the trivial rotator for a conventional fixed-point format is shown in Fig. 5(a). In this case, the word length of the data is kept, so no truncation or rounding is carried out. The rotation by 0° is a multiplication by 1, which does not modify the data. The rotation by −90° is a multiplication by $-j$. As $(x+jy)\cdot(-j) = y - jx$, the real part of the output, X_{Or}, is equal to the imaginary part of the input, X_{Bi}, whereas the imaginary part of the output, X_{Oi}, is the result of changing the sign of the real part of the input, X_{Br}. The sign change requires calculating the two's complement of the value, which comprises a bit-wise inversion plus adding 1 ULP. Additionally, the figure includes two multiplexers to select between the rotation by 0° or −90°

2) HUB: When the numbers are represented in HUB format, the trivial rotator is implemented as shown in Fig 5(b). In this case, the sign change is accomplished simply with a bit-wise inversion, as Section II-A explains. Thus, adding 1 ULP is not required, simplifying the logic of the trivial rotator for the HUB case.

Note that the word length is not modified in fixed-point and HUB trivial rotators, so they do not lead to any accuracy loss.

E. Adaptation of the Output From HUB to Conventional Format

As discussed at the beginning of Section III, we consider that the inputs and outputs of the FFT use the conventional fixed-point format.

When the FFT uses the HUB format, its output must be transformed into a conventional fixed-point number. This conversion is carried out at the output of the last butterfly by simply considering that the output value is a conventional truncated value instead of a rounded HUB one. The circuit for the last butterfly is the same as in other HUB stages, according to Fig. 3(b).

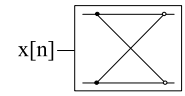
IV. Configurations Under Study

For this paper, we have analyzed many FFT architectures with different quantization schemes and number representations. The goal has been to improve the accuracy of the FFT while also having good results in terms of hardware resources and power consumption. Among all the cases that we have analyzed, Table I shows the most relevant configurations. The format of the inputs for each module is specified in the table as (HUB) for HUB input numbers and (-) for conventional ones. The adaptation from conventional numbers to HUB ones is done as explained in Section III-A. For the outputs of the modules, the quantization of conventional outputs can be by truncation (Trunc), rounding-half-up (Rnd), or keeping the word length (Exact). As trivial rotators keep the word length, no quantization occurs, so (-) is used for the outputs in the conventional format. For HUB outputs (HUB), only rounding-half-up by truncation is considered. The first and last butterflies are specified apart since they are special cases for HUB configurations.

The first column of Table I shows the acronyms of the configurations under study. The basic truncation case (BT) corresponds to the widely used configuration with conventional fixed-point numbers and truncation after each butterfly and rotator. This is the base case of our study. Analogously to BT, the basic rounding case (BR) uses rounding after all the butterflies and rotators of the architecture. BR makes half of the values exact, and the other half is rounded up, as they are exactly in the middle point. Consequently, in the butterflies, this rounding produces accuracy similar to BT but with the opposite bias.

In [22] and [23] truncation is alternated with rounding on each stage so that the negative and positive biases compensate each other. We use this approach in the TR1 configuration, where the outputs of general rotators and odd butterflies are rounded, and the outputs of even butterflies are truncated.

TABLE I
QUANTIZATION SCHEMES UNDER STUDY

Approach	First butterfly		Odd butterflies		Trivial rot.		Even butterflies		General rot.		Last butterfly (Even)		Previous Use
													
	In	Out	In	Out	In	Out	In	Out	In	Out	In	Out	
BT	-	Trunc	-	Trunc	-	-	-	Trunc	-	Trunc	-	Trunc	Widely used
BR	-	Rnd	-	Rnd	-	-	-	Rnd	-	Rnd	-	Rnd	Well known
TR1	-	Rnd	-	Rnd	-	-	-	Trunc	-	Rnd	-	Trunc	[22]
TR2	-	Rnd	-	Rnd	-	-	-	Exact	-	Trunc	-	Trunc	[23]
TR3	-	Trunc	-	Trunc	-	-	-	Exact	-	Rnd	-	Rnd	[23]
TR4	-	Rnd	-	Rnd	-	-	-	Trunc	-	Trunc	-	Trunc	No
TR5	-	Trunc	-	Trunc	-	-	-	Rnd	-	Trunc	-	Rnd	No
BHUB	-	HUB	HUB	HUB	HUB	HUB	HUB	HUB	HUB	HUB	HUB	Trunc	No
THUB	-	HUB	HUB	HUB	HUB	HUB	HUB	Trunc	-	HUB	HUB	Trunc	No

We also tested a similar configuration with odd butterflies truncated and even butterflies rounded. The results of these two configurations are almost the same, so we have only included the first one.

TR2 and TR3 are based on the description in [23], where each FFT stage alternates truncation and rounding. In TR2, the outputs of rotators are truncated, odd butterflies are rounded, and even ones are not quantized, except for the last one, which is truncated. TR3 is the opposite option, with rounding in rotators and last butterflies and truncation in odd butterflies.

TR4 is obtained by keeping the exact configuration of the butterflies as in TR1 but truncating the output of general rotators instead of rounding them. The opposite butterfly configuration, i.e., odd butterflies truncated and even butterflies rounded, is considered in TR5.

For the HUB format, the basic HUB configuration (BHUB) uses all the HUB circuits described in Section III so that all internal values between the modules are HUB numbers. The fact that the HUB format produces an actual rounding-half-up when truncating significantly improves accuracy without substantial hardware cost. However, similarly to the conventional case, in the butterflies, this rounding equals the accuracy of a truncation but with the opposite bias, since only one bit is discarded. This prevents the basic HUB implementation from reaching the level of accuracy of the best configurations with conventional numbers.

For this reason, we propose a second configuration (THUB), inspired by the alternate-stage rounding approach. THUB combines HUB and conventional numbers, intending to reduce logic and improve accuracy simultaneously. In this configuration, all internal butterflies use identical circuits, but odd ones are considered to have HUB output, whereas even ones are considered to have conventional ones. Moreover, all butterfly inputs are HUB values except for the first one. Thanks to this configuration, inputs to trivial rotators are always HUB numbers, significantly simplifying its implementation (see Section III-D). In contrast, inputs of the general rotators are always conventional numbers, reducing the multipliers' size. Moreover, all rounding is carried out by truncation, which also simplifies the circuit. Regarding accuracy, alternating the format at the output of the butterflies produces that truncation is alternated with rounding-half-up, whereas general rotator outputs are always rounded-half-up. This is the same rounding configuration as TR1 but with much less hardware cost, as shown in the experimental results.

Besides the architectures presented above, we have tested other options using HUB that have resulted in worse results. One of them is the use of HUB unbiased rounding in the butterflies. This rounding only requires a little more logic to zero the LSB of the output if the discarded bit equals zero [37]. However, the final SQNR was very similar (but worse) to the regular HUB version, so we do not recommend its use for this application.

Another modification to BHUB that we considered was for the values between the even butterflies and the general rotators. In the HUB version, the $WL+1$ bits of the butterfly outputs are truncated to get WL-bit values. However, at the input of the rotator, the ILSB is appended to get $WL+1$ bits again. Thus, we thought that keeping the exact value from the output of the butterfly would not cost much and would improve the accuracy. As expected, this modification improves slightly the accuracy with no significant cost compared to the regular HUB implementation. However, it has less accuracy and more area than the THUB configuration and, therefore, is not included in the experimental results.

Finally, we also tested a modification of THUB with the symmetric configuration in the butterflies, i.e., even butterflies with HUB output and odd ones with conventional values, conventional trivial rotators, and HUB general ones. This configuration has worse area and power consumption than THUB and even, reduces the accuracy.

V. SETUP FOR THE EXPERIMENTS

The accuracy of the quantization schemes under study is calculated through the signal-to-quantization-noise ratio (SQNR). This figure of merit reflects the relation between the input signal and the quantization error introduced in the calculation of the FFT.

Fig. 6 shows the setup used to measure the SQNR of the proposed FFT configurations. We have considered random

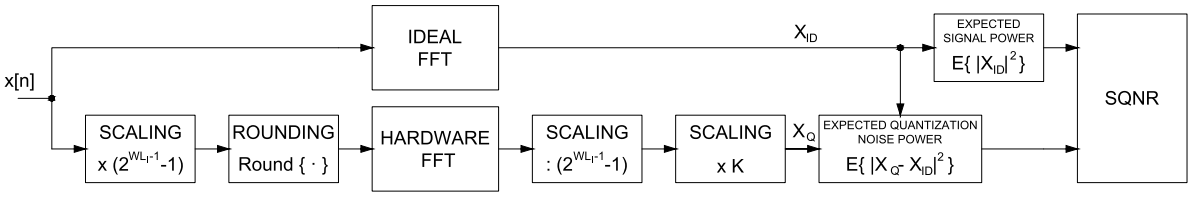


Fig. 6. Setup used to measure the SQNR of the proposed architectures.

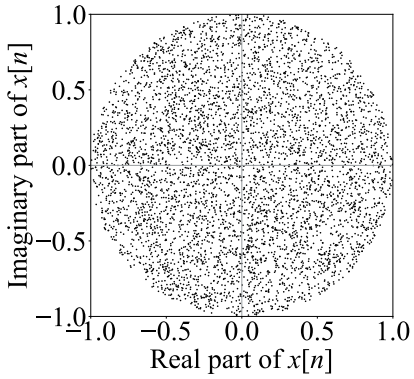


Fig. 7. Example of the distribution of input data, $x[n]$, with 5120 samples.

signals for the real and imaginary parts of $x[n]$. Fig. 7 shows an example of the statistical distribution of the input data in Fig. 6. The horizontal axis corresponds to the real part of $x[n]$, and the vertical axis corresponds to the imaginary part of $x[n]$. Note that data follow a uniform distribution strictly inside the circle with a radius equal to one. The reason for using this distribution is that input values whose magnitude is greater than or equal to one can cause overflow problems along the stages of the FFT. Note that if a twiddle factor rotates a value outside the circle, it may end up out of the square, causing overflow. Therefore, all numbers in the region outside the circle are not considered valid input data. Another alternative that we considered is to use only values in a square ranging from -0.5 to 0.5, both for the real and imaginary parts. This alternative reports SQNR values that are 3 dB smaller than the values reported in this paper. This is consistent with the fact that each extra bit increases the SQNR by 6 dB, as using the square in -0.5 to 0.5 represents a reduction of half a bit in the maximum magnitude of the inputs concerning the circle with magnitude 1.

Input data are generated with Matlab, where we calculate the ideal FFT of $x[n]$, X_{ID}, and prepare the inputs to be processed in hardware. The inputs are scaled by a factor $2^{WL-1}-1$ to cover the bit range of the word length, WL, and rounded to the nearest integer. These values are input to Vivado, where the FFT architectures are simulated. The simulation outputs are loaded again into Matlab. Then, these outputs are divided by $2^{WL-1}-1$ to compensate for the scaling of the input data before the simulation and, later, multiplied by a factor K. The truncation at the output of the butterflies divides the input of the next stage by two. As there are 10 stages and one butterfly at each stage, the output has to be multiplied by $K = 2^{10}$ to compensate for the scaling in butterflies. This way, the quantized signal, X_Q, maintains the same magnitude as X_{ID}. With the values of X_{ID} and X_Q, the SQNR is calculated as

$$\text{SQNR (dB)} = 10 \cdot \log_{10}\left(\frac{E\left\{|X_{ID}|^2\right\}}{E\left\{|X_Q - X_{ID}|^2\right\}}\right). \quad (9)$$

The numerator $E\{|X_{ID}|^2\}$ represents the estimated value of the power of the ideal signal and $E\{|X_Q - X_{ID}|^2\}$ represents the estimated value of the noise power. Each SQNR calculation considers 1000 trials, which means to simulate 1000 FFTs.

Finally, it is worth noting that the approach we use to calculate the SQNR may differ from the way it is calculated in other papers, leading to different values of SQNR in different works for the same FFT configuration. Due to this, contrary to other previous works in the literature that do not detail the setup to calculate the SQNR, we have provided a detailed explanation of our setup so that experiments from different works can be compared in the future.

VI. Experimental Results

For the experimental results, we have implemented the 1024-point radix-2^2 SDF FFT using all the configurations presented in Table I on a Virtex-7 XC7VX330TFFG1157-3 field-programmable gate array (FPGA) using Vivado 2022.1. The word length of all architectures is 16 bits for the real part and 16 bits for the imaginary part of the data. Table II shows the post-implementation results. General rotators are implemented with a complex multiplier that uses 3 DSP slices according to [38] to reduce power consumption.

By comparing the architectures in Table II, it can be observed that all of them use 4 BRAMs and 12 DSPs. However, the number of LUTs, FFs, and Slices differ. To calculate the power consumption under the same conditions, the power has been calculated for all the architectures at a frequency $f_{\text{CLK}} = 360$ MHz. However, some architectures can achieve a higher maximum clock frequency, f_{MAX}. Finally, the last column of the table shows the SQNR of the architectures, which has been calculated as explained in Section V.

For a more thorough comparison of the configurations under study, Table III shows the improvements concerning the conventional approach (BT) for all the figures of merit where the configurations differ, i.e., LUTs, FFs, Slices, f_{MAX}, power consumption, and SQNR. All the improvements are reported in percentage except for the SQNR improvement, which is reported in dB. Note also that the values in the table report improvements, not variations. For instance, if the area is reduced, the improvement is positive, not negative. This way, all the positive values in the table improve the

TABLE II
EXPERIMENTAL RESULTS FOR A 16-BIT RADIX-2^2 1024-POINT SDF FFTS USING DIFFERENT FORMATS

Parameter	BT	BR	TR1	TR2	TR3	TR4	TR5	BHUB	THUB
LUTs	1849	1871	1865	1910	1915	1857	1859	1747	1722
FFs	1292	1584	1594	1342	1606	1451	1292	1372	1292
Slices	585	632	652	589	671	596	577	557	537
BRAM	4	4	4	4	4	4	4	4	4
DSP	12	12	12	12	12	12	12	12	12
f_{CLK}(MHz)	360	360	360	360	360	360	360	360	360
f_{MAX}(MHz)	435	360	360	435	360	420	420	450	435
Latencia (μs)	2.95	2.95	2.95	2.95	2.95	2.95	2.95	2.95	2.95
Latencia (cyc.)	1063	1063	1063	1063	1063	1063	1063	1063	1063
Power (mW)	312	349	340	326	340	332	321	324	309
SQNR (dB)	56.25	57.15	61.42	61.17	61.44	61.07	60.00	60.09	61.20

TABLE III
IMPROVEMENTS IN AREA, POWER, f_{MAX} AND SQNR OF THE IMPLEMENTATIONS REPORTED IN TABLE II WITH RESPECT TO THE CONVENTIONAL APPROACH (BT)

Improvement in	BR	TR1	TR2	TR3	TR4	TR5	BHUB	THUB
LUTs (%)	-1.19	-0.87	-3.30	-3.56	-0.43	-0.54	+5.51	+6.87
FFs (%)	-22.60	-23.37	-3.87	-24.30	-12.31	0	-6.19	0
Slices (%)	-8.03	-11.45	-0.68	-14.70	-1.88	+1.37	+4.79	+8.21
f_{MAX} (%)	-17.24	-17.24	0	-17.24	-3.44	-3.44	+3.44	0
Power (%)	-11.85	-8.97	-4.48	-8.97	-6.41	-2.88	-3.85	+0.96
SQNR (dB)	+0.90	+5.17	+4.92	+5.19	+4.82	+3.75	+3.84	+4.95

figures of merit, whereas all the negative ones worsen them, which serves to show clearly what gets better and what gets worse.

In Table II, it can be observed that all the configurations improve the SQNR with respect to BT in the range from 0.90 to 5.19 dB. However, in BR, TR1, TR2, TR3, and TR4, this improvement comes at the cost of worsening the area, maximum frequency, and power consumption. In fact, the number of FFs is 3.87% to 24.30% worse and the power consumption is 4.48% to 11.85% worse, which is a considerable cost for the benefit in terms of SQNR.

TR1 and TR3 are the configurations that obtain the highest SQNR among all the approaches. This is achieved thanks to the alternation of rounding and truncation in the FFT stages, and the use of rounding for the general rotator outputs. However, this comes at the cost of having the highest area (including the number of FFs, slides, and LUTs) and power consumption. These increases are primarily due to the rounding on the general rotator and the exact output in the even butterflies. A better-balanced result among configurations with conventional format is obtained in the case of TR5, where there is the alternation of rounding and truncation in butterflies and the output of the general rotator is truncated. This configuration obtains a decent improvement of 3.75 dB in SQNR, while the other figures of merit only experience a slight variation.

Nonetheless, the best results are achieved by the approaches based on HUB. BHUB reaches the highest maximum clock frequency among all configurations while improving the area and the SQNR simultaneously. The only drawbacks are a 3.85% worse power consumption and an increase in the number of FFs.

Among all the approaches, the best one is the THUB configuration since all the figures of merit are improved or kept equal concerning BT. The alternation of truncation and rounding, along with the rounding (carried out by truncation) performed in the output of the general rotators, allows it to achieve a high SQNR, only 0.24 dB below TR3. However, in contrast to TR3, it also achieves the lowest area and power consumption, and the second-best speed among all configurations. Comparing THUB with TR1, which has almost the same SQNR as TR3 but less area, THUB uses almost 20% less FF and slides, almost 10% less LUTs and power and it also may work at 20% more speed. Thus, THUB is an excellent configuration for calculating the FFT, as it achieves a win-win deal that improves multiple figures of merit with respect to BT without worsening the values of any of them.

VII. INFLUENCE OF OTHER FFT PARAMETERS IN THE SQNR

Apart from the quantization schemes, other FFT parameters have an influence on the SQNR. In this section, we review the main parameters related to the FFT and the impact that they have on the SQNR. These parameters are:

- **Architecture type:** The FFT architecture itself does not have any influence on the SQNR. Note that different architectures differ in the order of the data, parallelization, or calculation of the FFT in the pipeline or iteratively. However, this does not affect the mathematical operations that are carried out, and therefore, the architecture type does not have any impact on the SQNR.
- **FFT size (N):** The SQNR depends on the FFT size. Larger FFTs have smaller SQNR. The SQNR for different FFT sizes is related in such a way that doubling N results

TABLE IV
SQNR DIFFERENCES IN DECIBELS OF TYPICAL FFT ALGORITHMS WITH RESPECT TO RADIX-2 DIF FOR DIFFERENT FFT SIZES AND $WL = 16$

Algorithm		N				
Radix	Dec.	64	128	256	512	1024
2	DIT	-0.82	-0.97	-1.09	-1.17	-1.23
2^2 or 4	DIF	0.02	0.04	0.03	0.00	0.05
2^2 or 4	DIT	-0.14	-0.19	-0.20	-0.24	-0.27
2^3 or 8	DIF	-0.10	-0.29	-0.10	-0.02	-0.02
2^3 or 8	DIT	-0.02	-0.06	-0.06	-0.15	-0.10
2^4 or 16	DIF	-0.11	-0.08	0.00	-0.25	-0.10
2^4 or 16	DIT	0.04	0.04	0.00	0.00	-0.02

in an SQNR that is approximately 3 dB smaller, as can be deduced from [17] and [24].

- **Parallelization (P):** Being related to the architecture type, parallelization of the architecture has no influence on the SQNR, as it does not affect the mathematical operations that are carried out.
- **Word length (WL):** Increasing the word length by one bit leads to a 6 dB higher SQNR. This comes from the fact that increasing one bit corresponds to multiplying the amplitude of the signal, A, by 2, while keeping the noise level. Thus, the difference in dB between a signal with amplitude $2A$ and a signal with amplitude A is:

$$\Delta \text{ dB } = 10 \cdot \log_{10}\left(\frac{(2A)^2}{A^2}\right) \approx 6 \text{ dB}. \tag{10}$$

- **Calculations in butterflies and rotators:** As butterflies and rotators are the components that carry out the mathematical operations, any modification on how these calculations are carried out affects the SQNR. One clear example is to substitute the rotators based on complex multipliers that we have used in the paper with the CORDIC rotator [39]. This would change the operations and the quantization, leading to new results for SQNR.
- **FFT algorithm:** FFT algorithms differ in the rotations at the FFT stages [40]. In decimation in frequency (DIF) algorithms, rotations are moved toward the first stages, whereas in decimation in time (DIT) algorithms, rotations are moved toward the last stage. These movements change the operations in the FFT. However, the impact of changing the algorithm in the SQNR is small. Table IV shows the difference in SQNR for various FFT algorithms with respect to the typical radix-2 DIF algorithm. The results consider different FFT sizes, N, and $WL = 16$ throughout the entire FFT. Note also that the mathematical operations in any radix-2^k algorithm are the same as in a radix-r algorithm with $r = 2^k$, as was shown in [40]. By analyzing the table, it can be observed that the magnitude of the SQNR difference is close to zero in most cases and only exceeds 1 dB in the case of the radix-2 DIT algorithm.
- **Clock frequency, throughput, and latency:** The accuracy of the FFT computations is independent of the speed at which these calculations are obtained. Thus, parameters such as the clock frequency, throughput, or latency of the design do not have any influence on the SQNR.
- **Area and number of components:** In the same way that the type of architecture does not have any influence on the SQNR, the area and number of components used to implement the FFT do not affect the SQNR.
- **Device:** The implementation of an FFT on an FPGA or application-specific integrated circuit (ASIC) has an impact on multiple parameters of the FFT. However, the mathematical calculations are the same in any device where the FFT is implemented. Therefore, the SQNR is unaffected by the device where the FFT is implemented.

As a result, the parameters that influence the SQNR are the quantization scheme used in the FFT architecture, the FFT size, the word length of the data, the calculations in butterflies and rotators, and the FFT algorithm. Other characteristics of the FFT, such as architecture type, parallelization, clock frequency, throughput, latency, area, number of components, and device, do not have any influence on the SQNR.

VIII. COMPARISON

Table V compares the SQNR of the proposed THUB architecture to other state-of-the-art FFT hardware architectures that report SQNR. As the mathematical computations are independent of the architecture, the table combines different types of FFT architectures: MDC, SDF, MSC, and MB. The table also includes architectures implemented on ASICs and FPGAs. For architectures on ASICs, the table reports the technology, voltage, and area. For architectures implemented on FPGAs, the table reports the type of FPGA, Virtex Ultrascale+ (VU+) or Virtex 7 (V7), and number of slices, LUTs, FFs, DSP slices, and BRAMs. The table also includes the FFT size, parallelization, word length, radix, quantization scheme, clock frequency, throughput, latency, and power consumption.

As can be observed, the architectures reported in Table V are very heterogeneous. The differences in N, P, architecture type, and device do not allow for a direct comparison of figures of merit such as throughput, area, latency, and power consumption. However, as explained in Section VII, the SQNR is independent of many characteristics related to the FFTs. This allows us to compare the SQNR of these architectures.

The SQNR is reported at the bottom of the table. To compare the SQNR values under similar circumstances, the last row of the table provides the equivalent SQNR (ESQNR), which removes the impact of the FFT size and the word length of the architectures. Thus, the equivalent SQNR is calculated as

$$\begin{aligned}\text{ESQNR (dB)} &= \text{SQNR (dB)} \\ &\quad + 3\log_2\left(\frac{N}{1024}\right) - 6(WL - 16),\end{aligned} \tag{11}$$

and corresponds to the SQNR that an equivalent FFT with $N = 1024$ and $WL = 16$ would have.

By comparing the equivalent SQNR in previous approaches to the results achieved by the proposed THUB implementation, it can be observed that the proposed design reaches the highest SQNR value, which highlights the value of the proposed quantization schemes toward improving the FFT accuracy.

As the ESQNR is only an approximation, we should also compare the proposed approach with other FFT architectures

TABLE V
COMPARISON OF STATE-OF-THE-ART FFT ARCHITECTURES

Approach	[31]	[30]	[32]	[33]	[35]	[15]	Prop.
Architecture	MDC	SDF	MDC	MSC	MB	MB	SDF
FFT size (N)	1024	1024	128	1024	4096	4096	1024
Parallelization (P)	4	1	4	4	64	16	1
Word length (WL)	16	16	13	16	16	21	16
Radix	2^5	2	8	2^5	2	8	2^2
Quantization Scheme	BT	BT	BT	BT	BT	BT	THUB
Device	ASIC	FPGA	ASIC	ASIC	FPGA	ASIC	FPGA
Technology (nm)	55	-	45	40	-	12	-
Voltage (V)	0.9	-	1.1	0.9	-	0.88	-
Area (mm^2)	0.21	-	0.22	0.18	-	0.75	-
FPGA Board	-	VU+	-	-	V7	-	V7
Slices	-	-	-	-	2319	-	537
LUTs	-	2305	-	-	6800	-	1722
FFs	-	1821	-	-	7027	-	1292
DSP Slices	-	27	-	-	96	-	12
BRAM	-	2	-	-	32	-	4
f_{CLK} (MHz)	320	500	402	330	300	1000	360
Throughput (MS/s)	1280	500	1608	1320	1476	3100	360
Latency (μs)	0.83	2.17	3.82	-	2.70	-	2.95
Power (mW)	17.02	490	36.13	30.90	1754	941.60	309
SQNR	40.30	54.43	39.40	40.58	44.80	82.69	61.20
Equivalent SQNR	40.30	54.43	48.40	40.58	50.80	58.69	61.20

that have the same FFT size and word length as the proposed one. Thus, if we consider architectures with 16 bits and 1024 points, the proposed design achieves 20.9, 6.81, and 20.62 dB higher SQNR than [30], [31], and [33], respectively. Therefore, the proposed approach not only achieves the highest equivalent SQNR but also significantly improves previous FFT architectures with similar characteristics in terms of accuracy.

IX. CONCLUSION

In this paper, we have analyzed several quantization schemes to improve accuracy in FFT architectures. Among them, alternatives that use a conventional number representation and alternate quantization and rounding along FFT stages improve accuracy at the cost of increasing area and power consumption. The best results are obtained for the HUB format combined with an alternating quantization strategy. This approach not only increases the SQNR but also reduces the area and power consumption, which is a win-win solution that improves many figures of merit simultaneously without worsening any of them. As a result, this approach is excellent for designing advanced FFT architectures.

REFERENCES

[1] S. Agarwal, S. R. Ahamed, A. Gogoi, and G. Trivedi, "A 28-Gbps radix-16, 512-point FFT processor-based continuous streaming OFDM for WiGig," *Circuits, Syst., Signal Process.*, vol. 41, no. 5, pp. 2871–2897, May 2022.

[2] A. Madanayake et al., "Fast radix-32 approximate DFTs for 1024-beam digital RF beamforming," *IEEE Access*, vol. 8, pp. 96613–96627, 2020.

[3] Z. Wei, H. Qu, W. Jiang, K. Han, H. Wu, and Z. Feng, "Iterative signal processing for integrated sensing and communication systems," *IEEE Trans. Green Commun. Netw.*, vol. 7, no. 1, pp. 401–412, Mar. 2023.

[4] V. Manuel 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, Reg. Papers*, vol. 70, no. 10, pp. 3992–4003, Oct. 2023.

[5] K. Virkler et al., "DSN radio astronomy spectrometer," in *Proc. 34th Gen. Assem. Sci. Symp. Int. Union Radio Sci. (URSI GASS)*, Aug. 2021, pp. 1–4.

[6] H. Kanders, T. Mellqvist, M. Garrido, K. Palmkvist, and O. Gustafsson, "A 1 million-point FFT on a single FPGA," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 66, no. 10, pp. 3863–3873, Oct. 2019.

[7] S. Corda, B. Veenboer, A. J. Awan, A. Kumar, R. Jordans, and H. Corporaal, "Near memory acceleration on high resolution radio astronomy imaging," in *Proc. 9th Medit. Conf. Embedded Comput. (MECO)*, Jun. 2020, pp. 1–6.

[8] L. Li and A. M. Wyrwicz, "Parallel 2D FFT implementation on FPGA suitable for real-time MR image processing," *Rev. Sci. Instrum.*, vol. 89, no. 9, pp. 1–9, Sep. 2018.

[9] J. Zhang et al., "Improved dynamic contrast-enhanced MRI using low rank with joint sparsity," *IEEE Access*, vol. 10, pp. 121193–121203, 2022.

[10] B. L. West, J. A. Fessler, and T. F. Wenisch, "Jigsaw: A slice-and-dice approach to non-uniform FFT acceleration for MRI image reconstruction," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. (IPDPS)*, May 2021, pp. 714–723.

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

[12] K. Möller, M. Kumm, M. Kleinlein, and P. Zipf, "Reconfigurable constant multiplication for FPGAs," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 36, no. 6, pp. 927–937, Jun. 2017.

[13] P. Paz, "Design space exploration of FFT architectures using rotator allocation," M.S. thesis, Dept. of Electron. Eng., Universidad Politécnica de Madrid, Madrid, SpAIN, 2020.

[14] M. Garrido and P. Malagón, "The constant multiplier FFT," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 68, no. 1, pp. 322–335, Jan. 2021.

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

[16] T. Lenart and V. Owall, "Architectures for dynamic data scaling in 2/4/8K pipeline FFT cores," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 14, no. 11, pp. 1286–1290, Nov. 2006.

[17] D. Guinart, "Deterministic analysis of the accuracy in FFT hardware architectures," M.S. thesis, Dept. Elect. Eng., Linkoping Univ., Linkoping, Sweden, 2012.

[18] M. Alrwashdeh and Z. Kollár, "Analysis of quantization noise in FFT algorithms for real-valued input signals," in *Proc. 32nd Int. Conf. Radioelektronika (RADIOELEKTRONIKA)*, Apr. 2022, pp. 1–5.

[19] N. K. Unnikrishnan, M. Garrido, and K. K. Parhi, "Effect of finite word-length on SQNR, area and power for real-valued serial FFT," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2019, pp. 1–5.

[20] M. Garrido, O. Gustafsson, and J. Grajal, "Accurate rotations based on coefficient scaling," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 58, no. 10, pp. 662–666, Oct. 2011.

[21] M. Garrido, F. Qureshi, and O. Gustafsson, "Low-complexity multiplierless constant rotators based on combined coefficient selection and shift-and-add implementation (CCSSI)," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 61, no. 7, pp. 2002–2012, Jul. 2014.

[22] P. Kabal and B. Sayar, "Performance of fixed-point FFT's: Rounding and scaling considerations," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, vol. 11, Apr. 1986, pp. 221–224.

[23] B. Zhou, Y. Peng, and D. Hwang, "Pipeline FFT architectures optimized for FPGAs," *Int. J. Reconfigurable Comput.*, vol. 2009, no. 1, pp. 1–9, Sep. 2009.

[24] T. Spiteri, "Compensating for bias due to rounding for fixed-point FFT," in *Proc. Eur. Conf. Commun. Syst. (ECCS)*, May 2023, pp. 1–5.

[25] S. D. Muñoz and J. Hormigo, "Improving fixed-point implementation of QR decomposition by rounding-to-nearest," in *Proc. Int. Symp. Consum. Electron. (ISCE)*, Jun. 2015, pp. 1–2.

[26] J. Hormigo and J. Villalba, "Optimizing DSP circuits by a new family of arithmetic operators," in *Proc. 48th Asilomar Conf. Signals, Syst. Comput.*, Nov. 2014, pp. 871–875.

[27] J. Hormigo and J. Villalba, "Measuring improvement when using HUB formats to implement floating-point systems under round-to-nearest," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 24, no. 6, pp. 2369–2377, Jun. 2016.

[28] J. Hormigo and J. Villalba, "HUB floating point for improving FPGA implementations of DSP applications," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 64, no. 3, pp. 319–323, Mar. 2017.

[29] J. Villalba-Moreno and J. Hormigo, "Floating point square root under HUB format," in *Proc. IEEE Int. Conf. Comput. Design (ICCD)*, Nov. 2017, pp. 447–454.

[30] V. M. Bautista and M. Garrido, "An automatic generator of non-power-of-two SDF FFT architectures for 5G and beyond," in *Proc. 38th Conf. Design Circuits Integr. Syst. (DCIS)*, Nov. 2023, pp. 61–66.

[31] M. Garrido, S.-J. Huang, and S.-G. Chen, "Feedforward FFT hardware architectures based on rotator allocation," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 65, no. 2, pp. 581–592, Feb. 2018.

[32] K. H. Viglianco, D. R. Garcia, and J. J. W. Kunst, "Implementation of a 4-parallel 128-point radix-8 FFT processor via folding transformation," in *Proc. Argentine Conf. Electron. (CAE)*, Mar. 2023, pp. 13–18.

[33] G.-T. Deng, M. Garrido, S.-G. Chen, and S.-J. Huang, "Radix-2k MSC FFT architectures," *IEEE Access*, vol. 11, pp. 81497–81510, 2023.

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

[35] Z. Kaya and M. Garrido, "Low-latency 64-parallel 4096-point memory-based FFT for 6G," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 70, no. 10, pp. 4004–4014, Oct. 2023.

[36] J. Hormigo and J. Villalba, "New formats for computing with real-numbers under round-to-nearest," *IEEE Trans. Comput.*, vol. 65, no. 7, pp. 2158–2168, Jul. 2016.

[37] J. Villalba-Moreno, J. Hormigo, and S. González-Navarro, "Unbiased rounding for HUB floating-point addition," *IEEE Trans. Comput.*, vol. 67, no. 9, pp. 1359–1365, Sep. 2018.

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

[39] J. E. Volder, "The CORDIC trigonometric computing technique," *IRE Trans. Electron. Comput.*, vol. 8, no. 3, pp. 330–334, Sep. 1959.

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

Mario Garrido (Senior Member, IEEE) received the M.Sc. and Ph.D. degrees in electrical 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, where he holds a Ramón y Cajal Research Fellowship. 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 and 2023.

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

Alejandro Portas received the bachelor's degree in industrial electronic engineering from the University of Córdoba (UCO) in September 2019 and the master's degree in electronic systems engineering (MUISE) from the Universidad Politécnica de Madrid (UPM) in July 2021, where he carried out the research for this paper.

After finishing his studies, he worked as an Embedded Software Engineer with Indra and Airbus, in air traffic management (ATM) and defense fields, respectively. Since September 2023, he has been with Sener in defense and space.

Javier Hormigo received the M.Sc. and Ph.D. degrees in telecommunication engineering from the Universidad de Málaga, Spain, in 1996 and 2000, respectively. He joined the Universidad de Málaga in 1997, where he is currently a Full Professor with the Computer Architecture Department. He has published six patents and over 60 papers in international journals and conferences, winning the "Best Paper Award" at the ICCD 2013 Conference. His research interests include designing accelerators, with a specific focus on high-level synthesis for FPGA and the application of non-conventional arithmetic in these application-specific circuits. He also serves as an Associate Editor and the Topical Editor for IEEE TRANSACTIONS ON COMPUTERS and has been recognized with the "IEEE TC Award for Editorial Service and Excellence" in 2021 and 2023.