A Pipelined FFT Architecture for Real-Valued Signals

Mario Garrido, Keshab. K. Parhi, *Fellow, IEEE*, and J. Grajal

***Abstract*—This paper presents a new pipelined hardware architecture for the computation of the real-valued fast Fourier transform (RFFT). The proposed architecture takes advantage of the reduced number of operations of the RFFT with respect to the complex fast Fourier transform (CFFT), and requires less area while achieving higher throughput and lower latency.**

The architecture is based on a novel algorithm for the computation of the RFFT, which, contrary to previous approaches, presents a regular geometry suitable for the implementation of hardware structures. Moreover, the algorithm can be used for both the Decimation In Time (DIT) and Decimation In Frequency (DIF) decompositions of the RFFT and requires the lowest number of operations reported for radix 2.

Finally, as in previous works, when calculating the RFFT the output samples are obtained in a different order. The problem of reordering these samples is solved in this paper and a pipelined circuit that performs this reordering is proposed.

***Index Terms*—Fast Fourier Transform (FFT), Real-Valued Signals, Pipelined Architecture, Reordering Circuit, Decimation-in-Time, Decimation-in-Frequency, Memory Reduction**

I. Introduction

The fast Fourier transform (FFT) is one of the most important algorithms in the field of digital signal processing, used to efficiently compute the discrete Fourier transform (DFT). For the computation of the FFT, pipelined hardware architectures [1]-[9] are widely used because they offer high throughput and low latency as well as a reasonably low area and power consumption. This makes them attractive for a large variety of applications, specially when they present real-time requirements. Thus, in order to provide solutions to present and future applications, hardware designers keep improving the signal processing capabilities of pipelined architectures of the FFT.

The FFT internally operates over complex numbers and previous works offer efficient designs for the computation of the FFT of complex input samples (CFFT). However, they are not optimized for the computation of the FFT of real input samples (RFFT). Indeed, when the input samples are real the spectrum of the FFT is symmetric [10] and approximately half of the operations are redundant [11].

The RFFT plays an important role in different real-time applications. In medical applications such as ECG (Electrocardiography) [12] or EEG (Electroencephalography) [13], the power spectral density (PSD) of various real-valued signals has to be estimated. This requires calculation of the RFFT repetitively on many overlapping windows of the signals, where a specialized hardware implementation can make use of a higher speed clock to meet real-time constraints. Moreover, in implantable or portable devices [12], [13], a dedicated hardware can save power consumption. On the other hand, the RFFT is a key element in technologies based on DMT (discrete multitone) modulation, such as ADSL (Asymetric Digital Subscriber Line) or VDSL (Very high bit-rate Digital Subscriber Line) [14], [15]. Nowadays, high signal processing capabilities are required for second-generation standards (ADSL2/2+ [16] and VDSL2 [17]). Moreover, DMT has been used at rates of 24 Gb/s in local area networks (LAN) [18]. Finally, very high performance is also necessary in digital wideband receivers [19], [20]. Currently, the RFFT has to be computed in real time over a dataflow of 2 GSamples/s [20].

Thus, in order to meet the increasing demand on real-time capabilities of new applications, much research has been carried out on pipelined architectures of the CFFT. On the other hand, for those applications with real input signals, a dedicated pipelined RFFT architecture can lead to savings in area and power consumption, while offering high signal processing capabilities. However, to the best of our knowledge no hardware-efficient pipelined architecture for the computation of the RFFT based on the Cooley-Tukey algorithm [21] has been proposed yet. There exist only some pipelined architectures [22], [23] based on the Bruun algorithm [24]. However, the Bruun algorithm has not been widely adopted since it was demonstrated [25] that the noise is significantly higher than that in the Cooley-Tukey algorithm [21].

The lack of specific pipelined architectures for the RFFT is due to the fact that the specific algorithms proposed for the computation of the RFFT [11], [26]-[30], do not lead to regular geometries, which are necessary for designing pipelined architectures. These specific approaches describe programs based on removing the redundancies of the CFFT when the input is real, and can be used to efficiently compute the RFFT in a DSP (Digital Signal Processor) or in in-place architectures [31]. A memory-based or in-place architecture consists of a memory and a processing unit. The data are loaded, processed and stored again in the memory until all the operations of the algorithms are performed. This kind of architecture allows the design of circuits with low area and power consumption, but it is not suitable for real-time applications.

There exist other techniques described in [32], [33], which take advantage of the CFFT to calculate the RFFT. On one hand, the *doubling algorithm* uses the CFFT to compute

Mario Garrido and J. Grajal are with the Department of Signal, Systems and Radiocommunications, Unversidad Politécnica de Madrid, 28040 Madrid, Spain, e-mails: mgarrido@gmr.ssr.upm.es, jesus@gmr.ssr.upm.es

Keshab K. Parhi is with the Department of Electrical and Computer Engineering, University of Minnesota, Minneapolis, MN 55455, USA, e-mail: parhi@umn.edu

two RFFTs simultaneously. On the other hand, the *packing algorithm* forms a complex sequence of length $N/2$ taking the even and odd indexed samples of the real input sequence of length N, and calculates the $N/2$-point CFFT of the complex sequence. In these CFFT-based techniques, additional operations are necessary to obtain the final results from the outputs of the CFFT. The packing algorithm has been used to implement some in-place architectures [15], [34]. In these architectures the hardware of the CFFT can be reused for computing the post-processing stage. However, no pipelined architecture has been proposed, not only because some of the adders and multipliers saved in the CFFT need to be used for post-processing, but mainly because the samples need to be reordered before the post-processing stage [15], increasing the memory and complicating the control.

Both in the specific algorithms for the computation of the RFFT and in the CFFT-based ones, the output samples are provided in different orders [27], which are different from the bit-reversal one of the CFFT [35]. The sorting of the outputs of the RFFT is also a problem not solved in the literature so far.

In this work we propose the first pipelined architecture for the computation of the RFFT based on the Cooley-Tukey algorithm. It combines the advantages of the pipelined architectures with the reduction of operations achieved by the specific algorithms. This is possible due to the proposed algorithm for the computation of the RFFT which solves the irregularities of the RFFT. Moreover, this approach is valid for both Decimation In Time (DIT) and Decimation In Frequency (DIF) decompositions and is generalizable for any number of points N, which is power of 2. Furthermore, the problem of the output order of the samples is solved and a pipelined circuit that performs the reordering is proposed.

In the next section we briefly review the RFFT and the existing techniques to compute it. In Section III we develop the algorithm that allows the design of regular hardware architectures for the computation of the RFFT, and the novel proposed pipelined architecture is presented Section IV. Next, in Section V the architecture is evaluated and compared to previous approaches, and some conclusions are drawn in Section VI. Finally, the reordering of the output samples and the proposed solution are discussed in Appendix A.

II. The RFFT

The N-point DFT of a sequence $x[n]$ is defined as:

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

where $W_N^{nk} = e^{-j\frac{2\pi}{N}nk}$.

The RFFT considers the input sequence, $x[n]$, to be a real sequence, i.e., $\forall n,\ x[n] \in \mathbb{R}$. It is easy to demonstrate [10] that if $x[n]$ is real, then the output $X[N-k]$ is complex conjugate of $X[k]$. Consequently, an RFFT can be considered a conventional FFT with the additional conditions:

$$Im(x[n]) = 0 \tag{2}$$

and

$$X[N-k] = X^*[k] \tag{3}$$

These two conditions that distinguish the CFFT and the RFFT do not lead, however, to a direct simplification of equation (1) for the RFFT, and the specific algorithms for the computation of the RFFT require complicated developments. Thus, other simpler techniques that use the CFFT to compute the RFFT are sometimes preferred. Next, these approaches are reviewed.

A. Computation of the RFFT using the CFFT

1) Direct use of the CFFT: The first idea when it is necessary to compute an FFT over a real input signal is to use the CFFT. As the real numbers are a subset of the complex ones, the trivial solution is to set the imaginary part of the input to zero. Although this procedure does not make an efficient use of the resources, it is very simple and it is not necessary to modify the CFFT. Indeed, it is the solution adopted for many if not all real-time applications.

2) Doubling Algorithm: Another alternative is to take advantage of the CFFT to simultaneously compute the FFT of two real signals $x_1[n]$ and $x_2[n], n = 0, 1, \ldots, N-1$ as it is explained in [32], [33]. This process requires to form the signal $y[n] = x_1[n] + j \cdot x_2[n]$ and use an N-point CFFT to obtain $Y[k] = X_1[k] + j \cdot X_2[k]$. It is important to notice that both $X_1[k]$ and $X_2[k]$ are complex, so they cannot be obtained directly from $Y[k]$. Therefore, $2(N-1)$ additions are required to separate the outputs, in addition to the operations of the CFFT.

3) Packing Algorithm: Given a real signal $x[m]$, $m = 0, 1, \ldots, M-1$ it is also possible to compute the RFFT using an $M/2$-point CFFT [32], [33]. This technique is sometimes called *packing algorithm* because it takes the odd and even indexed samples of the signal and form the complex signal $y[n] = x[2n] + j \cdot x[2n+1]$, $n = 0, 1, \ldots, N-1$ and $N = M/2$. Then the N-point CFFT is applied to obtain $Y[k], k = 0, 1, \ldots, N-1$. As in the *doubling algorithm*, $2(N-1)$ additions are required to separate the outputs of the CFFT. Moreover, in the *packing algorithm* it is necessary to include an additional stage to compute the outputs of the RFFT, which requires $4N-1$ extra additions and $4(N-1)$ multiplications.

B. Specific Algorithms for the Computation of the RFFT

The greater reduction in the number of operations is obtained by using the specific algorithms for the computation of the RFFT. Most of them are obtained from the CFFT by applying the properties of the RFFT in order to remove the redundant operations. The first proposed algorithms were defined for the DIT (Decimation In Time) decomposition of the FFT. The DIT FFT has the property that the samples at each intermediate stage can be computed using the conventional FFT [10]. Consequently, equation (3) can be applied at each stage, and only one half of the intermediate outputs must be calculated, whereas the rest can be obtained by conjugating those intermediate values.

This is the basic idea of algorithms proposed for split-radix [11], [26], radix-2 [27], [30] and high radices [28]. These algorithms are, however, not valid for the DIF (Decimation In Frequency) decomposition of the FFT because it is not possible to apply the property (3) at each stage. On the other hand, it has been also demonstrated that it is possible to obtain the same savings for the DIF decomposition using an alternative algorithm [29] that makes use of linear-phase sequences.

In general, the number of multiplications in all of these algorithms is reduced to half of that required for the CFFT, and the number of additions is $N-2$ less than half the additions of the CFFT. Likewise, only half of the memory is needed. Thus, there are only slight differences among all of them in the number of operations and in the order in which the computations are performed.

III. Proposed Algorithm for the Computation of the RFFT

A. Basis of the algorithm

Figure 1 shows the flow graph of an N-point FFT for the case of $N=16$, radix $r=2$, and decomposed according to the decimation in frequency (DIF) [10]. The graph is divided into $log_r N = 4$ stages and each of them consists of a set of butterflies and rotators. The numbers at the input and the output of the graph represent respectively the index of the input and output samples, whereas each number, ϕ, in between the stages indicates a rotation by:

$$e^{-j\frac{2\pi}{N}\phi} \tag{4}$$

If we consider that the inputs are complex, all the internal nodes and outputs of the graph are needed for the computation of the CFFT, and the regularity of the flow graph leads to efficient pipelined architectures [2]. On the other hand, if the inputs are real, it is possible to simplify the graph according to the properties of the RFFT, as explained next.

1.- The first simplification is to consider that in the real FFT $X[N-k] = X^*[k]$. According to this, $N/2-1$ outputs of the FFT are redundant and can be removed. Most approaches [11], [27], [31] obtain either the frequencies with indexes $k=[0,N/2]$ or $k=[0,N/4]\cup[N/2,3N/4]$ and, if necessary, calculate the rest of the frequencies by conjugating these results. However, considering that $k' = N/4-k-1$ for the indices $k,k' = 0,\ldots,N/4-1$ and using the property (3):

$$X[4k+3] = X^*[N-4k-3] = X^*[4k'+1] \tag{5}$$

leads to a more efficient architecture.

Consequently, the set of frequencies $X[4k+3]$ can be obtained by conjugating frequencies $X[4k+1]$, as mentioned in [26] for the split-radix RFFT. According to Figure 1, samples $X[4k+3]$ are the last quarter of the outputs; so all the butterflies used exclusively to compute these samples may be removed, which are represented by the lower darkened region.

The same concept can be applied to samples $X[8k+6]$, which can be computed by conjugating samples $X[8k+2]$. Generalizing this idea, samples $X[2^\alpha\cdot(4k+3)]$ can be computed by conjugating the samples $X[2^\alpha\cdot(4k+1)]$, where

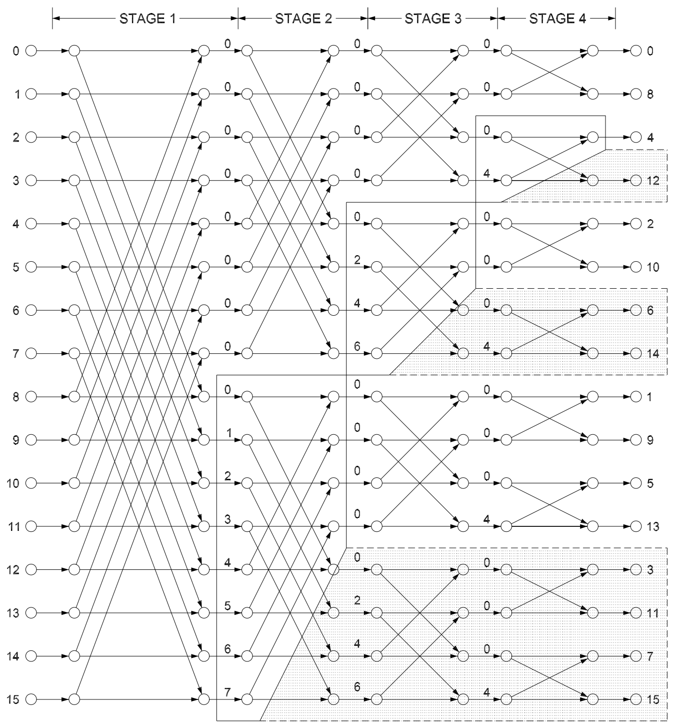


Fig. 1. Flow graph of a 16-point DIF FFT. The darkened regions and the boxed components are considered in the simplifications of the new algorithm for the RFFT.

$k=0,\ldots,N/(4\cdot 2^\alpha)-1$, for all $\alpha = 0,\ldots,log_2 N-2$. Thus, all the darkened regions in Figure 1 can be removed and only $N/2-1$ outputs of the FFT need to be computed.

2.- The second simplification refers to the fact that $Im(x[n]) = 0$. According to this, every piece of data is real until it is rotated for the first time. In Figure 1, all the additions performed before the data reach the boxed components are real and, thus, the number of additions in that area is halved with respect to the CFFT. Once the data reach the first rotations, the data are necessarily complex until the end of the FFT.

3.- One further simplification of the boxed components may be carried out. Let's assume that the first stage of the FFT is $s=1$ and the last one is $s=n\equiv log_r N$, and X_s are the outputs of stage s. According to this and Figure 1, in the second stage it is not necessary to compute the data $X_2[i+3N/4]$, $i=0,\ldots,N/4-1$, and samples $X_2[i+N/2]$ are calculated as:

$$\begin{aligned} X_2[i+N/2] &= X_1[i+N/2]\cdot e^{-j\frac{2\pi}{N}i} + \\ &+X_1[i+3N/4]\cdot e^{-j\frac{2\pi}{N}(i+N/4)} \end{aligned} \tag{6}$$

This calculation requires 2 rotations and 1 addition. However, expanding the expression and taking into account that both $X_1[i+N/2]$ and $X_1[i+3N/4]$ are real samples, we can obtain:

$$X_2[i+N/2] = \{X_1[i+N/2] - j\cdot X_1[i+3N/4]\}\cdot e^{-j\frac{2\pi}{N}i} \tag{7}$$

The resulting expression only requires one rotation and no additions because $X_1[i+N/2]$ and $X_1[i+3N/4]$ are real and rotations by 1, -1, j and $-j$ are trivial, taking into account that they can be calculated by interchanging the real and imaginary

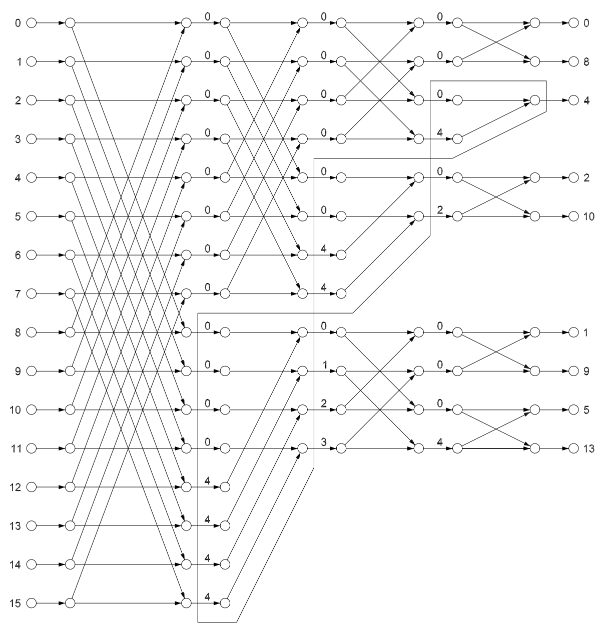


Fig. 2. Simplified flow graph of a 16-point FFT for real input samples, before a regular structure is obtained.

parts and/or changing the sign of the data. This simplification can be extended to all the boxed components, which reduces even more the number of rotators and adders of the FFT.

Figure 2 represents the flow graph of the RFFT once the explained simplifications have been carried out. All the data previous to the boxed components are real, and only real butterflies are necessary for this region. On the other hand, according to equation (7) the boxed components do not require any operation but the trivial rotation $-j$, since the additions that appear in the graph do not have to be performed because one of the inputs is real and the other one is purely imaginary. Finally, the complex rotations appear after the boxed components and the butterflies need to be complex. Consequently, it is easy to see from Figure 2 that the total number of real additions is $N \log_2 N - (N-2)$ or, considering that $n = log_2 N$:

$$\# \; add = (n-1) \cdot 2^n + 2 \tag{8}$$

Likewise, the number of non-trivial complex multiplications can be obtained from the flow graph as:

$$\# \; mult = (n-4) \cdot 2^{n-2} + n \tag{9}$$

With regard to the output samples, all the frequencies $X[0]$ to $X[N/2]$ can be computed from the obtained data by conjugating those frequencies with index greater than $N/2$.

The explained simplifications can also be applied to the DIT FFT. In this case it is first necessary to redraw the typical flow graph of the DIT FFT so that the inputs are in natural order and the outputs in bit-reversal [36]. The result only differs from that of Figure 2 in the rotations performed at the different stages. In this case, the number of non-trivial complex multiplications can be calculated as:

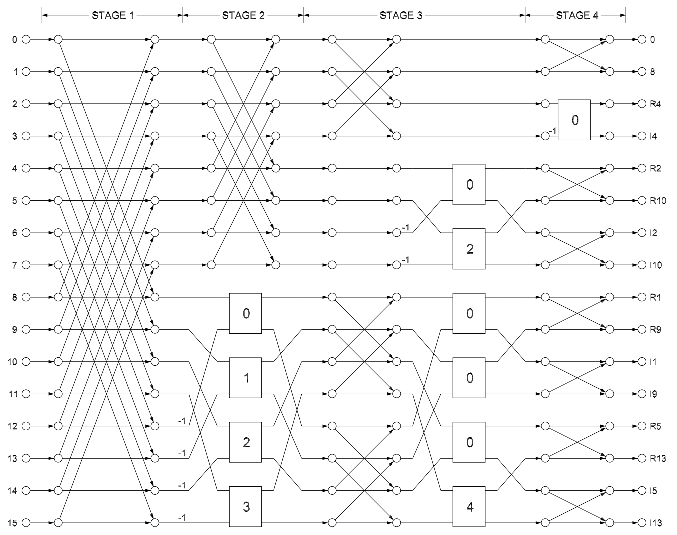


Fig. 3. Proposed flow graph of a 16-point DIF RFFT. All edges are real and the boxed numbers represent rotations of the RFFT according to equation (4). The sign of the data is changed in edges where -1 is depicted.

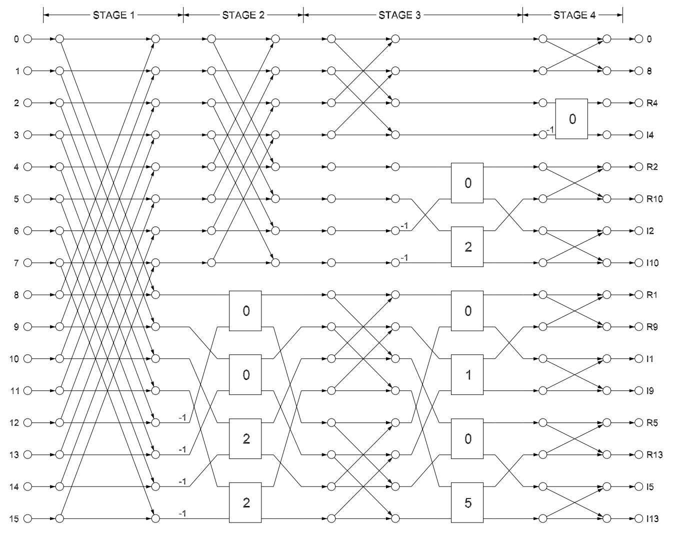


Fig. 4. Proposed flow graph of a 16-point DIT RFFT. All edges are real and the boxed numbers represent rotations of the RFFT according to equation (4). The sign of the data is changed in edges where -1 is depicted.

$$\# \; mult = (n-3) \cdot 2^{n-2} + 1 \tag{10}$$

Finally, as in the other specific designs of RFFT, the obtained structures are irregular because the symmetries of the CFFT have been used to reduce the number of operations. Although the flow graph in Figure 2 could be used to implement an in-place architecture, it is not suitable for the implementation of a pipelined architecture.

B. Obtaining regularity

The last step of the algorithm consists of transforming the flow graph in Figure 2 to obtain a flow graph with regular geometry. The obtained structure is depicted in Figure 3 for the DIF FFT, whereas the result for the DIT decomposition is

TABLE I
NUMBER OF OPERATIONS OF THE ALGORITHMS FOR THE COMPUTATION OF THE RFFT.

	RFFT Algorithm		Radix	Real Multiplications	Real Additions
Specific Algorithms	Sorensen [11]	DIT	Split-radix	$(2n-6)\cdot 2^{n-2}+2$	$(n-1)\cdot 2^{n}+2$
	Duhamel [26]	DIT	Split-radix	$(2n-6)\cdot 2^{n-2}+2$	$(n-1)\cdot 2^{n}+2$
	Bergland [27]	DIT	radix-2	$(3n-10)\cdot 2^{n-2}+4$	$(n-1)\cdot 2^{n}+2$
	Sekhar [29]	DIF	radix-2	$(3n-10)\cdot 2^{n-2}+4$	$(n-1)\cdot 2^{n}+2$
	Sundararajan [30]	DIT	radix-2	$(11/2n-25/3)\cdot 2^{n-2}+10/3$	$(n-1)\cdot 2^{n}+2$
	Proposed	DIF	radix-2	$(3n-13)\cdot 2^{n-2}+4n-2$	$(n-1)\cdot 2^{n}+2$
		DIT	radix-2	$(3n-10)\cdot 2^{n-2}+4$	$(n-1)\cdot 2^{n}+2$
CFFT-based	Packing Algorithm		split-radix	$2n\cdot 2^{n-2}$	$(n+2)\cdot 2^{n}+3$
			radix-2	$(3n-5)\cdot 2^{n-2}+4$	$(n+2)\cdot 2^{n}+3$
	Doubling Algorithm		split-radix	$(4n-12)\cdot 2^{n-2}+4$	$(2n+2)\cdot 2^{n}-2$
			radix-2	$(6n-20)\cdot 2^{n-2}+8$	$(2n+2)\cdot 2^{n}-2$

shown in Figure 4. Contrary to Figures 1 and 2, all the edges in Figures 3 and 4 are real, i.e, the data have been separated into their real and imaginary components. Consequently, in the flow graph the rotations have two inputs and two outputs. They are represented by the boxes with a number inside, which indicate the rotation angles according to equation (4). The upper inputs and outputs are used for the real part of the samples and the lower ones for the imaginary part. On the other hand, those trivial rotations by $-j$ in the boxed components of Figure 2 only operate over a real sample, which is equivalent to multiply it by -1 and input it in the imaginary part of the rotator, as it is done in Figures 3 and 4. Finally, the output samples of the flow graph include a letter to indicate if the value corresponds to the real part (R) or the imaginary part (I) of the output. Moreover, the complex butterflies after the boxed components in Figure 2 have been unfolded in the new structure.

Contrary to the flow graph in Figure 2, in the structures of Figures 3 and 4 every stage has N inputs and outputs and all of them are real samples. Moreover, at every stage butterflies operate over samples whose indexes differ the same quantity and non-trivial rotations are placed in particular positions. These regularities allow the development of efficient pipelined hardware architectures, as explained in section IV. On the other hand, it can be noticed that, as in the other specific algorithms of the RFFT and the CFFT-based ones, the outputs are not available in bit-reversed order. However, it is possible to sort them out as explained in Appendix A.

C. Comparison to other algorithms for the computation of the RFFT

Table I compares the number of operations of the different approaches for the computation of the RFFT and, consequently, their efficiency when they are implemented in DSPs. The number of real multiplications has been calculated according to the criterion used in previous approaches [11]: Rotations by $45°$,$135°$,$225°$ and $315°$ require 2 real multiplications and the rest of non-trivial rotations are calculated with three real multiplications. On the other hand, the additions in Table I include those of the butterflies and those required for the post-processing in the CFFT-based algorithms.

As it is shown, the number of operations in all the approaches has the same order of magnitude. Among them, the specific algorithms based on split-radix require less operations than those based on radix-2. The same is true for CFFT-based algorithms. On the other hand, for the same radix, the specific algorithms are better than the packing algorithm both in the number of multiplications and additions. Likewise, they are also better than the doubling algorithm. This is not only because for the doubling algorithm two signals need to be computed in parallel, but also because it needs twice the number of multiplications and more than twice the number of additions.

Comparing the specific algorithms for the computation of the RFFT, the proposed approach is the only one that can be used for both the DIT and DIF decompositions of the FFT. Moreover, the DIF version of the algorithm has the lowest number of operations reported for radix-2. The reason why the DIF version has less operations than the DIT one is due to the fact that the simplification of Equation (7) can only be applied to the DIF decomposition.

However, the most relevant advantage of the proposed algorithm over the previous approaches is that it offers a regular structure suitable for the implementation of hardware architectures. It may be noted that a regular structure for RFFT with fewest number of operations has not been presented so far.

IV. PIPELINED ARCHITECTURE OF THE RFFT

Figure 5 shows the proposed pipelined architecture for a 16-point DIF RFFT, obtained from the graph of Figure 3. It is a radix-2 feedforward pipelined structure that maximizes the use of the multipliers, as well as achieves a throughput of 4 samples per clock cycle. As in the flow graph, all the edges carry real samples. Therefore, the radix-2 butterflies $(R2)$ process two real inputs and, thus, they only consist of a real adder and a real subtractor. Likewise, the rotators $(\otimes)$ have two inputs and two outputs, as in the flow graph. The upper input receives the real part of the sample to be rotated and the lower one the imaginary part. Finally, the rest of circuits are shuffling structures used for reordering the samples according to the dataflow. They are composed of buffers and

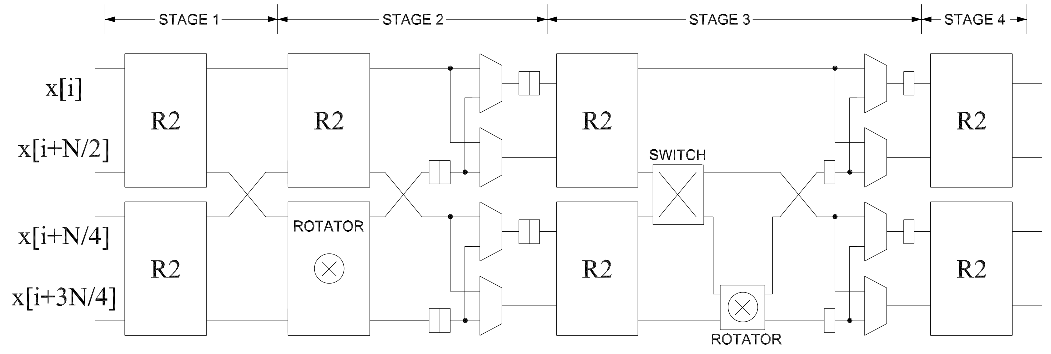


Fig. 5. Proposed pipelined architecture for the computation of the 16-point DIF RFFT. The architecture is composed of radix-2 real butterflies ($R2$), rotators ($\otimes$), and shuffling structures, which include buffers and multiplexors.

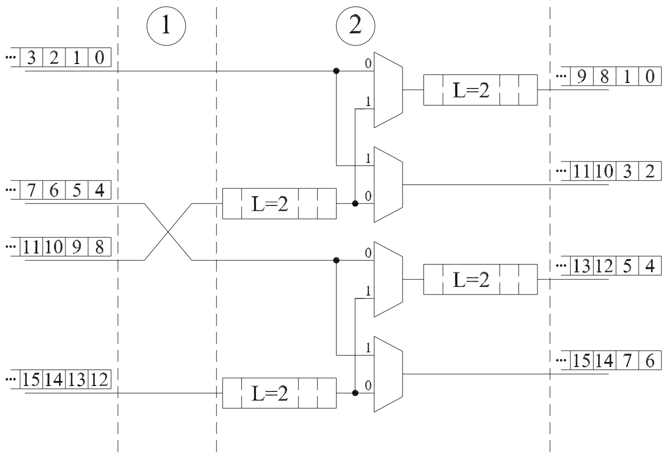


Fig. 6. Shuffling structure of the pipelined RFFT. This examples shows the shuffling of the second stage of a 16-point RFFT.

multiplexors. In a general case of an N-point RFFT, with N power of two, the circuit requires $2{\cdot}log_2(N)-1$ real butterflies, $log_2(N)-2$ rotators, and buffers or memories of a total size $N-4$.

Considering both Figures 3 and 5, the first stage computes only real butterflies. According to the input order of the data, the upper butterfly of the structure computes the pairs of samples $(0,8)$, $(1,9)$, $(2,10)$ and $(3,11)$, whereas the lower butterfly operates samples $(4,12)$, $(5,13)$, $(6,14)$ and $(7,15)$.

On the other hand, both in the architecture and in the flow graph, the upper outputs of the butterflies of the first stage are connected to a butterfly of the second stage, whereas the lower outputs are rotated in the second one. Previous to the rotator, it is necessary to compute a trivial rotation of -1, which can be embedded in the rotator.

At the second stage, the butterflies operate over each pair of samples $X_1[i]$ and $X_1[i{+}N/4]$, and the rotations are calculated over $X_1[i+N/2]$ and $X_1[i+3N/4]$, for $i=0,\dots,3$. Consequently, the indexes of two samples operated together always differ by $N/4$. On the other hand, at the third stage, the indexes of two samples operated together differ by $N/8$.

According to this, the shuffling structure depicted in Figure 6, which corresponds to the second stage of the architecture of Figure 5, shows how the order of the data required at the second stage of the 16-point RFFT is transformed to the order of the third stage. The structure consists of buffers and multiplexors. In the Figure, the length of the buffers is $L=2$, and the numbers of the inputs and the outputs represent the index of each piece of data. Thus, the structure receives in parallel samples from the second stage, whose indexes differ by $N/4=4$, and provides in parallel samples adapted to the third stage, where the indexes differ by $N/8=2$.

The shuffling is performed in two steps. The first one interchanges the intermediate inputs and the second step interleaves the data. Considering the upper circuit of the second step, indexed samples $(0,1,2,3)$ and $(8,9,10,11)$ are received respectively at each of the inputs. Initially the control of the multiplexors is set to "0" and, thus, samples with indexes $(0,1)$ are stored in the output buffer and $(8,9)$ in the input one. Next, the control signal switches to "1" and indexed samples $(0,1)$ are provided at the output in parallel with $(2,3)$, whereas $(8,9)$ pass to the output buffer and $(10,11)$ are stored in the input one. These samples will be provided in parallel at the output when the multiplexor switches again to "0".

In a general case of an N-point RFFT, the shuffling structure at a stage $s\in[2,n{-}1]$ requires buffers of length $L=N/2^{s+1}$, and, as can be observed in Figure 5, for the first stage of the RFFT, $s=1$, only the first step of the shuffling structure is required.

On the other hand, the third stage of the architecture in Figure 5 includes a switch before the rotator. This switch is necessary in every stage $s\in[3,n-1]$. As it is shown in Figure 3 the rotations required by the boxed components of Figure 1 operate over samples whose indexes differ the same quantity as the butterflies at the same stage. For these rotations, the switch does not swap the inputs. However, for the rest of rotations the switch is activated and the samples that come from the lower output of the upper butterfly are routed through the rotator.

Finally, Figure 7 represents the hardware architecture for the DIT RFFT. It only differs from the DIF structure in Figure 5 on the placement of the boxed switch and the rotator, due to the fact that the positions of the rotations in the flow graphs of Figures 3 and 4 are different.

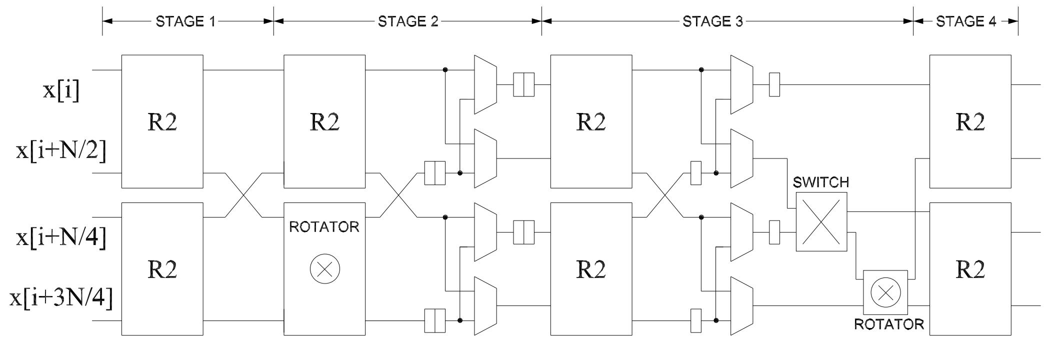


Fig. 7. Proposed pipelined architecture for the computation of the 16-point DIT RFFT. The architecture is composed of radix-2 real butterflies ($R2$), rotators ($\otimes$), and shuffling structures, which include buffers and multiplexors.

TABLE II
COMPARISON OF PIPELINED HARDWARE ARCHITECTURES FOR THE COMPUTATION OF AN N-POINT RFFT.

PIPELINED ARCHITECTURE	AREA				PERFORMANCE	
	Complex Rotators	Complex Adders	Complex Memory		Latency (cycles)	Throughput (samples/cycle)
			Samples	Rotations		
FB Radix 2, [1]	log_4N-1	$2(log_4N)$	$4N/3$	N	N	1
FB Radix 2, [2]	$2(log_4N-1)$	$4(log_4N)$	N	N	N	1
FB Radix 4, [2], [3]	log_4N-1	$8(log_4N)$	N	N	N	1
FB Radix 2^2, [2]	log_4N-1	$4(log_4N)$	N	N	N	1
FB Split-radix, [4]	log_4N-1	$4(log_4N)$	N	N	N	1
FF Radix 2, [5]	$2(log_4N-1)$	$2(log_4N)$	N	N	N	1
FF Radix 2, [2]	$2(log_4N-1)$	$4(log_4N)$	N	N	$N/2$	2
FF Radix 2, [3]	$2(log_4N-1)$	$4(log_4N)$	$4N$	N	N	2
FF Radix 2, [6]	$2(log_4N-1)$	$4(log_4N)$	N	N	$N/2$	2
FF Radix 4, [7]	log_4N-1	$3(log_4N)$	$2N$	N	N	1
FF Radix 4, [8]	$3(log_4N-1)$	$8(log_4N)$	N	N	$N/4$	4
FF Radix 4, [3]	$3(log_4N-1)$	$8(log_4N)$	$8N/3$	N	$N/3$	4
FF Radix 4, [6]	$3(log_4N-1)$	$8(log_4N)$	N	N	$N/3$	4
Proposed	$2(log_4N-1)$	$4(log_4N)$	$N/2$	$3N/4$	$N/4$	4

V. COMPARISON AND ANALYSIS

Table II compares the proposed structure to other efficient pipelined architectures for the case of computing an N-point RFFT. The proposed design is the only specific approach for the computation of the RFFT and, thus, it takes advantage of the reduced number of operations required by the RFFT with respect to the CFFT. The other approaches are not specific for the RFFT and can be used to calculate the CFFT.

The table shows the trade off between area and performance. The area is measured in terms of the number of complex rotators, adders and memory addresses, whereas the performance is represented by the throughput and the latency. In all cases, the throughput is that for which each architecture has been designed, and the number of components and the latency are measured from the butterflies of the first stage to those of the last stage, i.e, circuits for reordering the input and output samples have not been considered. This criterion shows the lowest area and the highest performance that each architecture can obtain. As it can be observed in the table, feedback (FB) architectures and some feedforward (FF) ones [5], [7] require less hardware components and achieve a throughput of one sample per clock cycle. On the other hand, the parallelization of feedforward architectures has the advantage of a higher throughput and a lower latency due to an increase in area.

The proposed architecture is in the group of the feedforward ones. It uses radix-2 but can process 4 samples in parallel, achieving a higher performance than feedback designs, both in terms of latency and throughput. On the other hand, compared to other radix-2 feedforward architectures, the proposed design doubles the throughput and halves the latency and the data memory, while keeping the same number of rotators and adders. Therefore, although the new approach uses radix-2, it achieves the same throughput as other radix-4 feedforward architectures. Compared to these high-throughput architectures, the proposed one obtains a significant reduction in the number of rotators, adders and memory addresses.

The reduction in memory is an important advantage of the proposed architecture. It only needs a total of N real memory addresses for the samples or, equivalently, $N/2$ complex ones. This involves a great reduction in area, taking into account that, except in case of computing a low number of samples, the memory takes up most of the area of the circuit. Moreover, the memory used to store the coefficients for the rotations has been reduced to $3N/4$. Thus, the low area makes the architecture very suitable for the implementation of long-length transforms.

Finally, it is interesting to analyze the case that input samples arrive in natural order. Under these circumstances feedforward architectures require input memories for reordering the samples, whereas feedback structures do not need any additional hardware. For the proposed design it can be used a memory of N real samples or, equivalently, $N/2$ complex samples. As the throughput of the architecture is

4 samples per clock cycle, the memory will be filled in $N/4$ cycles, leading to a total latency of the circuit of $N/2$ cycles. Consequently, using the same memory as a feedback architecture, the proposed design can process 4 times more samples in half the time.

As a conclusion, the high performance and the low area of the new architecture make it very attractive for the computation of the RFFT in real-time applications.

VI. Conclusion

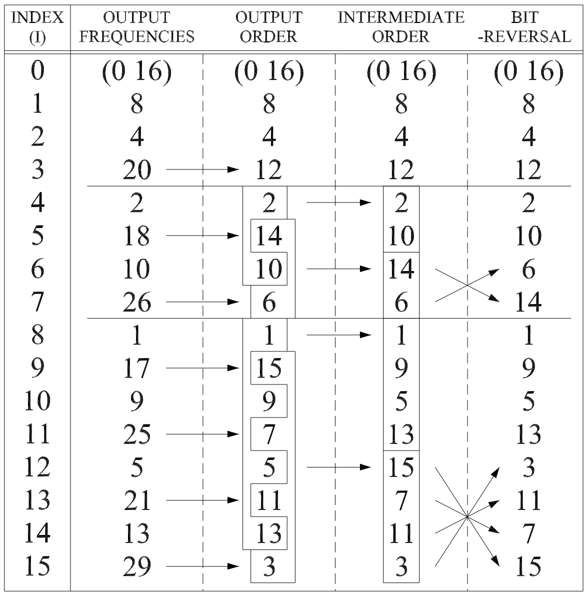
The previous approaches on the RFFT had demonstrated that it requires half the operations of the CFFT. This paper shows that this reduction of operations is not only theoretical but also can be applied to the design of efficient hardware architectures for the computation of the RFFT. This is possible due to the novel algorithm proposed in this paper for the computation of the RFFT, which obtains a regular geometry similar to that of the CFFT, and is valid for both the DIF and DIT decompositions of the RFFT. Based on this algorithm, a new pipelined architecture for the computation of the RFFT is presented. It processes 4 samples in parallel and requires significantly less memory than other efficient pipelined structures. The low area and latency, and the high throughput of the circuit make it attractive for the computation of the FFT in any application that processes real samples. Moreover, the circuit is very efficient for long-length transforms due to the reduction in memory, and it is also very suitable for real-time applications because of its high throughput capabilities.

Appendix A
Reordering of the output samples

The scrambled order of the output samples is an inherent problem of the FFT. In the CFFT the outputs are obtained in the so called bit-reversal order [35]. In in-place architectures it is possible to reuse the memory to sort out the output samples, which increases the latency and reduces the throughput of the system.

On the other hand, in pipelined architectures an extra memory of N addresses is necessary to perform the bit-reversal, which increases the area and the latency. Samples are stored in the memory in natural order using a counter for the addresses and then they are read in bit-reversal by reversing the bits of the counter. Moreover, it is possible to use this strategy for calculating the bit-reversal of a series of FFTs. In this case, the first FFT is stored in natural order. Next, the count is in bit-reversal and, thus, the frequencies are provided in the correct order, while the outputs of the second FFT are stored in the memory in bit-reversal order. Since the bit-reversal is an inversion operation, i.e, $Br(x) = Br^{-1}(x)$, the outputs of the second FFT must be read in natural order to sort out the frequencies. At the same time, the third FFT is stored in natural order, which completes the cycle.

Nevertheless, for reordering the outputs of the RFFT a more complicated algorithm is required. Next the algorithm is explained to be performed in-place and then it is shown that it is possible to perform it in pipeline and the corresponding circuit is presented.



INDEX (I)	OUTPUT FREQUENCIES	OUTPUT ORDER	INTERMEDIATE ORDER	BIT -REVERSAL
0	(0 16)	(0 16)	(0 16)	(0 16)
1	8	8	8	8
2	4	4	4	4
3	20	12	12	12
4	2	2	2	2
5	18	14	10	10
6	10	10	14	6
7	26	6	6	14
8	1	1	1	1
9	17	15	9	9
10	9	9	5	5
11	25	7	13	13
12	5	5	15	3
13	21	11	7	11
14	13	13	11	7
15	29	3	3	15

Fig. 8. Problem of the reordering of the output samples of the RFFT.

Figure 8 shows how to sort out the outputs of the RFFT to obtain them in bit-reversal order for the case of a 32-point RFFT, where only frequencies 0 to 16 are necessary.

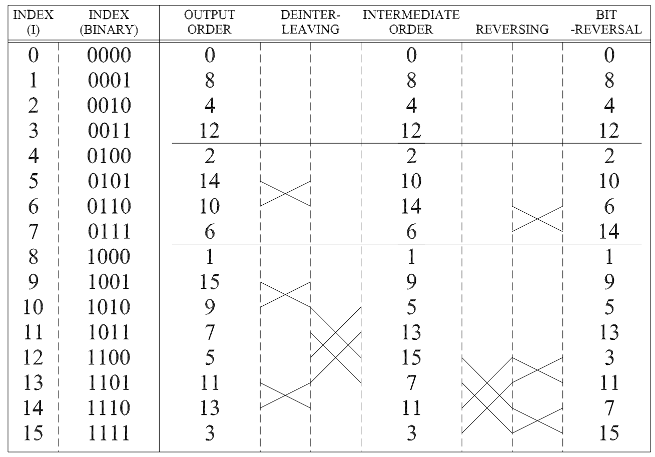
The first column (*index*) represents the order of arrival of the output samples. The second column (*output frequencies*) indicates the indexes of the output frequencies of the RFFT. Frequencies $k = 0$ and $k = N/2 = 16$ are both real and they are grouped together at the output of the RFFT. The goal is to obtain the frequencies in the order of the first column provided the order of the second column.

First of all, the data at frequencies greater than $N/2$ must be conjugated to obtain all the frequencies in the interval $k \in [0, N/2]$, which appear in the third column (*output order*).

Next, the output order given in the second column must be converted into bit-reversal order. To achieve this goal, it is first necessary to realize that the output order and the samples in bit-reversal have certain similarities: the order of the first four samples (indexes $I = 0, \dots, 3$) is the same, the following four ($I = 4, \dots, 7$) can be obtained by shuffling the samples in the same position, and the last half of the samples ($I = 8, \dots, 15$) can also be obtained by shuffling. It can be generalized for higher number of points: the bit-reversal order of all samples in positions $I \in [2^p, 2^{p+1} - 1]$, $p = 0, \dots, log_2 N - 2$ can be obtained by shuffling the samples in those positions according to the output order.

Given any of those sets of samples, the sorting can be performed in two stages. First, it is necessary to notice that the first and the second half of the samples in the sets are interleaved. According to this, the *intermediate order* shown in the fourth column of Figure 8 is obtained by deinterleaving these groups of data. Next, the bit-reversal order is obtained by reversing the order of the last half of the samples of each set.

Figure 9 shows the deinterleaving and reversing procedures for the case under study. Assuming that $b_{n-1}, \dots, b_1, b_0$ are



INDEX (I)	INDEX (BINARY)	OUTPUT ORDER	DEINTER-LEAVING	INTERMEDIATE ORDER	REVERSING	BIT -REVERSAL
0	0000	0		0		0
1	0001	8		8		8
2	0010	4		4		4
3	0011	12		12		12
4	0100	2		2		2
5	0101	14		10		10
6	0110	10		14		6
7	0111	6		6		14
8	1000	1		1		1
9	1001	15		9		9
10	1010	9		5		5
11	1011	7		13		13
12	1100	5		15		3
13	1101	11		7		11
14	1110	13		11		7
15	1111	3		3		15

Fig. 9. Solution to the reordering of the output samples of the RFFT.

the bits of the index I, the deinterleaving moves a sample in position $P = b_{n-1}, b_{n-2}, \dots, b_p, b_{p-1}, \dots, b_1, b_0$ to position $P' = b_{n-1}, b_{n-2}, \dots, b_p, b_0, b_{p-1}, \dots, b_2, b_1$, where $p = \lfloor log_2(P) \rfloor$. This is performed by successively interchanging the bits $b_\alpha, b_{\alpha+1}$ from $\alpha = 0$ to $\alpha = p - 2$. Note that in this figure $p = 2$ for $I \in [4, 7]$ and $p = 3$ for $I \in [8, 15]$, so each group of samples is treated differently.

On the other hand, the reversing moves the data in positions $P = b_{n-1}, b_{n-2}, \dots, b_p, 1, b_{p-2}, \dots, b_1, b_0$ to position $P' = b_{n-1}, b_{n-2}, \dots, b_p, 1, \bar{b}_{p-2}, \dots, \bar{b}_1, \bar{b}_0$. The reversing can be performed by negating the position bit by bit and interchanging the corresponding data.

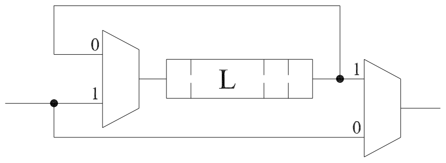


Fig. 10. Basic circuit for the reordering of the output data.

This procedure can be easily performed in place, but it is also possible to implement a pipelined hardware structure for the reordering of the samples. It is necessary to realize that every stage of the deinterleaving and the reversing interchanges samples in positions P and $P' = P + L$, where L is a constant. The simple circuit in Figure 10 performs this exchange. If the multiplexer is set to "1" the samples will be provided at the output in the same order as in the input, whereas setting it to "0" the input sample in position P' is forwarded to position P, while the data in position P is fed back to the buffer and will appear at the output in position P'.

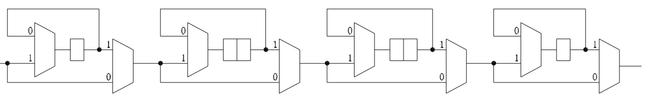


Fig. 11. Structure for the reordering of the output data of a 32-point RFFT.

Joining the stages of the deinterleaving and the reversing, the circuit in Figure 11 performs the reorder of the outputs to obtain them in bit-reversed order. As it can be seen, it only uses six complex registers for a 32-point RFFT. In a general case, a N-point RFFT requires buffers or a memory of $2 \cdot (N/8 - 1)$ complex data to obtain the outputs in bit-reversal, and a memory of $N/2$ complex data to obtain them in natural order.

References

[1] L. Yang, K. Zhang, H. Liu, J. Huang, and S. Huang. 'An efficient locally pipelined FFT processor'. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 53(7):585–589, July 2006.

[2] S. He and M. Torkelson. 'Design and implementation of a 1024-point pipeline FFT processor'. *Custom Integrated Circuits Conference, Santa Clara, CA*, pages 131–134, May 1998.

[3] M. A. Sánchez, M. Garrido, M. L. López, and J. Grajal. 'Implementing the FFT Algorithm on FPGA Platforms: A Comparative Study of Parallel Architectures'. *XIX International Conference on Design of Circuits and Integrated Sistems (DCSI 2004), (Bourdeaux, France)*, November 2004.

[4] Wen-Chang Yeh and Chein-Wei Jen. 'High-speed and low-power split-radix FFT'. *IEEE Transactions on Signal Processing*, 51(3):864–874, March 2003.

[5] Y.-N. Chang. 'An Efficient VLSI Architecture for Normal I/O Order Pipeline FFT Design'. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 55(12):1234–1238, December 2008.

[6] Chao Cheng and Keshab K. Parhi. 'High-Throughput VLSI Architecture for FFT Computation'. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 54(10):863–867, October 2007.

[7] G. Bi and E.V. Jones. 'A pipelined FFT processor for world-sequential data'. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 37(12):1982–1985, December 1989.

[8] E.E. Swartzlander, W.K.W. Young, and S.J. Joseph. 'A radix 4 delay commutator for fast Fourier transform processor implementation'. *IEEE Journal of Solid-State Circuits*, 19(5):702–709, October 1984.

[9] Y.-W. Lin and C.-Y. Lee. 'Design of an FFT/IFFT Processor for MIMO OFDM Systems'. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 54(4):807–815, April 2007.

[10] A.V.Oppenheim and R.W.Schafer. *Discrete-Time Signal Processing*. Prentice Hall, 1989.

[11] H.V. Sorensen, D.L. Jones, M.T. Heideman, and C.S. Burrus. 'Real-valued fast Fourier transform algorithms'. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 35:849–863, June 1987.

[12] M.-H. Cheng, L.-C. Chen, Y.-C. Hung, and C.M. Yang. 'A real-time maximum-likelihood heart-rate estimator for wearable textile sensors'. *Engineering in Medicine and Biology Society, 2008. EMBS 2008. 30th Annual International Conference of the IEEE*, pages 254–257, August 2008.

[13] R.F. Yazicioglu, P. Merken, R. Puers, and C. Van Hoof. 'Low-Power Low-Noise 8-Channel EEG Front-End ASIC for Ambulatory Acquisition Systems'. *European Solid-State Circuits Conference, ESSCIRC 2006*, pages 247–250, 2006.

[14] O.W. Ibraheem and N.N. Khamiss. 'Design and Simulation of Asymmetric Digital Subscriber Line (ADSL) Modem'. *ICTTA 2008*, pages 1–6, April 2008.

[15] Hsiang-Feng Chi and Zhao-Hong Lai. 'A cost-effective memory-based real-valued FFT and Hermitian symmetric IFFT processor for DMT-based wire-line transmission systems'. *IEEE International Symposium on Circuits and Systems*, 6:6006–6009, May 2005.

[16] ITU-T Recommendation G.992.5: Asymmetric Digital Subscriber Line (ADSL) transceivers Extended bandwidth ADSL2 (ADSL2plus). 2005.

[17] ITU-T Recommendation G.993.2: Very high speed digital subscriber line transceivers 2 (VDSL2). 2006.

[18] S.C.J. Lee et al. '24-Gb/s Transmission over 730 m of Multimode Fiber by Direct Modulation of an 850-nm VCSEL using Discrete Multi-tone Modulation'. *https://w3.tue.nl/fileadmin/ele/TTE/ECO/Files/Pubs_2007/Lee_PDP6_OFC_07.pdf*, 2007.

[19] G. Lopez-Risueno, J. Grajal, and A. Sanz-Osorio. 'Digital channelized receiver based on time-frequency analysis for signal interception'. *IEEE Transactions on Aerospace and Electronic Systems*, 41(3):879–898, July 2005.

[20] Y.-H.G. Lee and C.-I.H. Chen. 'Dual Thresholding for Digital Wideband Receivers with Variable Truncation Scheme'. *IEEE International Symposium on Circuits and Systems, ISCAS*, May 2009.

[21] J.W. Cooley and J.W. Tukey. 'An algorithm for the machine calculation of complex Fourier series'. *Math. Comput.*, 19:297–301, 1965.

[22] R. Storn. 'A novel radix-2 pipeline architecture for the computation of the DFT'. *IEEE International Symposium on Circuits and Systems*, 2:1899–1902, June 1988.

[23] Y. Wu. 'New FFT structures based on the Bruun algorithm'. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 38(1):188–191, January 1990.

[24] G. Bruun. 'z-Transform DFT filters and FFT's'. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 26(1):56–63, February 1978.

[25] R. Storn. 'Some results in fixed point error analysis of the Bruun-FTT algorithm'. *IEEE Transactions on Signal Processing*, 41(7):2371–2375, July 1993.

[26] P. Duhamel. 'Implementation of Split-radix FFT algorithms for complex, real, and real-symmetric data'. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 34(2):285–295, April 1986.

[27] G.D. Bergland. 'A Fast Fourier Transform Algorithm for real-valued Series'. *Commun. ACM*, 11(10):703–710, October 1968.

[28] G.D. Bergland. 'A Radix-eight Fast Fourier Transform subroutine for real-valued Series'. *IEEE Trans. on Audio and Electroacoustics*, AU-17(2):138–144, June 1969.

[29] B.R. Sekhar and K.M.M. Prabhu. 'Radix-2 decimation-in-frequency algorithm for the computation of the real-valued FFT'. *IEEE Transactions on Signal Processing*, 47(4):1181–1184, April 1999.

[30] D. Sundararajan, M.O. Ahmad, and M.N.S. Swamy. 'Some results in fixed point error analysis of the Bruun-FTT algorithm'. *IEEE Transactions on Signal Processing*, 41(7):2371–2375, July 1997.

[31] J.A. Hidalgo-López, J.C. Tejero, J. Fernández, E.Herruzo, and A. Gago. 'New architecture for RFFT calculation'. *Electronics Letters*, 30(22):1836–1838, October 1994.

[32] E. Oran Brigham. *The fast Fourier transform and its applications*. Prentice Hall, 1988.

[33] Winthrop W. Smith and Joanne M. Smith. *Handbook of Real-Time Fast Fourier Transforms*. Wiley-IEEE Press, 1995.

[34] A. Wang and A.P. Chandrakasan. 'Energy-aware Architectures for a Real-valued FFT Implementation'. *International Symposium on Low Power Electronics and Design*, pages 360–365, August 2003.

[35] Lawrence R. Rabiner and Bernard Gold. *Theory and Application of Digital Signal Processing*. Prentice Hall, 1975.

[36] Mario Garrido Gálvez. 'Desarrollo de algoritmos para detección de señales en tiempo real mediante FPGAs'. *ETSI de Telecomunicación, UPM*, December 2004.

Mario Garrido received his M.S. degree in Electrical Engineering from the Technical University of Madrid (UPM), Madrid, Spain, in December 2004. He is currently working towards the Ph.D degree in the Department of Signals, Systems and Radiocommunications, UPM.

His research interests include the design and optimization of VLSI architectures for signal processing applications.

Keshab K. Parhi (S'85-M'88-SM'91-F'96) received his B.Tech., MSEE, and Ph.D. degrees from the Indian Institute of Technology, Kharagpur, the University of Pennsylvania, Philadelphia, and the University of California at Berkeley, in 1982, 1984, and 1988, respectively. He has been with the University of Minnesota, Minneapolis, since 1988, where he is currently Distinguished McKnight University Professor in the Department of Electrical and Computer Engineering.

His research addresses VLSI architecture design and implementation of physical layer aspects of broadband communications systems, error control coders and cryptography architectures, high-speed transceivers, and ultra wideband systems. He is also currently working on intelligent classification of biomedical signals and images, for applications such as seizure prediction, lung sound analysis, and diabetic retinopathy screening. He has published over 450 papers, has authored the text book *VLSI Digital Signal Processing Systems* (Wiley, 1999) and coedited the reference book *Digital Signal Processing for Multimedia Systems* (Marcel Dekker, 1999).

Dr. Parhi is the recipient of numerous awards including the 2004 F.E. Terman award by the American Society of Engineering Education, the 2003 IEEE Kiyo Tomiyasu Technical Field Award, the 2001 IEEE W.R.G. Baker prize paper award, and a Golden Jubilee award from the IEEE Circuits and Systems Society in 1999. He has served on the editorial boards of the IEEE TRANSACTIONS ON CAS, CAS-II, VLSI Systems, Signal Processing, Signal Processing Letters, and Signal Processing Magazine, and served as the Editor-in-Chief of the IEEE Trans. on Circuits and Systems - I (2004-2005 term), and currently serves on the Editorial Board of the Journal of VLSI Signal Processing. He has served as technical program cochair of the 1995 IEEE VLSI Signal Processing workshop and the 1996 ASAP conference, and as the general chair of the 2002 IEEE Workshop on Signal Processing Systems. He was a distinguished lecturer for the IEEE Circuits and Systems society during 1996-1998. He was an elected member of the Board of Governors of the IEEE Circuits and Systems society from 2005 to 2007.

J. Grajal was born in Toral de los Guzmanes (Len), Spain, in 1967. He received his M.S and Ph.D. in Electrical Engineering degrees from the Technical University of Madrid (UPM), Madrid, Spain, in 1992 and 1998, respectively. Since 2001 he has been an Associate Professor at the Department of Signals, Systems, and Radiocommunications of the Technical School of Telecommunication Engineering of the same University.

His research activities are in the area of hardware-design for radar systems, radar signal processing and broadband digital receivers for radar and spectrum surveillance applications.