Abstract: The appearance of radix-$2^2$ was a milestone in the design of pipelined FFT hardware architectures. Later, radix $2^2$ was extended to radix $2^n$. However, radix-$n$ was only proposed for single-path delay feedback (SDF) architectures, but not for feedforward ones, also called multi-path delay commutator (MDC). This paper presents the radix $2^n$ feedforward (MDC) FFT architectures. In feedforward architectures radix $2^n$ can be used for any number of parallel samples which is a power of two. Furthermore, both decimation in frequency (DIF) and decimation in time (DIT) decompositions can be used. In addition to this, the designs can achieve very high throughputs, which make them suitable for the most demanding applications. Indeed, the proposed radix $2^n$ feedforward architectures require fewer hardware resources than parallel feedback ones, also called multi-path delay feedback (MDF), when several samples in parallel must be processed. As a result, the proposed radix $2^n$ feedforward architectures not only offer an attractive solution for current applications, but also open up a new research line on feedforward structures.

Keywords: Fast Fourier Transform (FFT), Multipath Delay Commutator (MDC), Pipelined Architecture, Radix-$2^n$, VLSI.

I. INTRODUCTION

The Fast Fourier transform (FFT) is one of the most important algorithms in the field of digital signal processing. It is used to calculate the discrete Fourier transform (DFT) efficiently. In order to meet the high performance and real-time requirements of modern applications, hardware designers have always tried to implement efficient architectures for the computation of the FFT. In this context, pipelined hardware architectures are widely used, because they provide high throughputs and low latencies suitable for real time, as well as a reasonably low area and power consumption. There are two main types of pipelined architectures: feedback (FB) and feedforward (FF). On the one hand, feedback architectures are characterized by their feedback loops, i.e., some outputs of the butterflies are fed back to the memories at the same stage. Feedback architectures can be divided into single-path delay feedback (SDF), which processes a continuous flow of one sample per clock cycle, and multi-path delay feedback (MDF) or parallel feedback, which process several samples in parallel. On the other hand, feedforward architectures also known as multi-path delay commutator (MDC) do not have feedback loops and each stage passes the processed data to the next stage.

These architectures can also process several samples in parallel. In current real-time applications, the FFT has to be calculated at very high throughput rates, even in the range of Gig samples per second. These high-performance requirements appear in applications such as orthogonal frequency division multiplexing (OFDM) and ultra wideband (UWB). In this context two main challenges can be distinguished. The first one is to calculate the FFT of multiple independent data sequences. In this case, all the FFT processors can share the rotation memory in order to reduce the hardware. Designs that manage a variable number of sequences can also be obtained. The second challenge is to calculate the FFT when several samples of the same sequence are received in parallel. This must be done when the required throughput is higher than the clock frequency of the device. In this case it is necessary to resort to FFT architectures that can manage several samples in parallel. As a result, parallel feedback architectures, which had not been considered for several decades, have become very popular in the last few years. Conversely, not very much attention has been paid to feedforward (MDC) architectures.

This paradoxical fact, however, has a simple explanation. Originally, SDF and MDC architectures were proposed for radix-$2^2$ and radix-$4$. Some years later, radix $2^n$ was presented for the SDF FFT as an improvement on radix-$2$ and radix-$4$. Next, radix $2^n$ and radix-$4$, which enable certain complex multipliers to be simplified, were also presented for the SDF FFT. Finally, the current need for high throughput has been met by the MDF, which includes multiple interconnected SDF paths in parallel. In this work we present the radix $2^n$ feedforward FFT architectures. The proposed design can be used for any number of parallel samples which is a power of two. Radix $2^n$ FFT architectures are shown to be more hardware-efficient than previous feedforward and parallel feedback designs in the literature. This makes them very attractive for the computation of the FFT in the most demanding applications.
The lower edges of the butterflies are always multiplied by -1. These -1 are not depicted in order to simplify the graphs. The numbers at the input represent the index of the input sequence, whereas those at the output are the frequencies of the output signal X[k]. Finally, each number, , in between the stages indicates a rotation by

\[ W^\phi_N = e^{-j \frac{2\pi}{N} \phi} \tag{2} \]

As a consequence, samples for which \( \Phi = 0 \) do not need to be rotated. Likewise, if \( \Phi \in \{0, N/4, N/2, 3N/4\} \) the samples must be rotated by \( 0^0, 270^0, 180^0, \) and \( 90^0 \), which correspond to complex multiplications by \( 1, -j, j, \) and \( j \), respectively. These rotations are considered trivial, because they can be performed by interchanging the real and imaginary components and/or changing the sign of the data.

Radix-2\(^2\) is based on radix-2\(^2\) and the flow graph of a radix-2 DIF FFT can be obtained from the graph of a radix-2 DIF one. This can be done by breaking down each angle \( \Phi \), at odd stages into a trivial rotation and a non-trivial one, \( \Phi' \), where \( \Phi = \Phi \mod N/4 \), and moving the latter to the following stage. This is possible thanks to the fact that in the radix-2 DIF FFT the rotation angles at the two inputs of every butterfly, \( \Phi_A \) and \( \Phi_B \), only differ by 0 or \( N/4 \). Thus, if \( \Phi_A = \Phi' \) and \( \Phi_B = \Phi + N/4 \), the rotation \( \Phi \) is moved to the following stage in accordance with

\[ A e^{-j \frac{2\pi}{N} \phi'} = B e^{-j \frac{2\pi}{N} (\phi'+N/4)} = [A \pm (-j)B] \cdot e^{-j \frac{2\pi}{N} \phi'} \tag{3} \]

Where the first side of (3) represents the computations using radix-2 and the second one using radix 2\(^2\), and \( B \) being the input data of the butterfly. In radix-2, A and B are rotated before the butterfly is computed, whereas in radix 2\(^2\) B is rotated by the trivial rotation \(-j\) before the butterfly, and the remaining rotation is carried out after the butterfly. Consequently, rotations by \( \Phi' \) can be combined with those rotations of the following

\[ A e^{-j \frac{2\pi}{N} \phi'} = B e^{-j \frac{2\pi}{N} (\phi'+N/4)} = [A \pm (-j)B] \cdot e^{-j \frac{2\pi}{N} \phi'} \]

Fig. 1. Flow graph of the 16-point radix-2 DIF FFT.

Fig. 2. Flow graph of the 16-point radix-2\(^2\) DIF FFT.

Fig. 3. Proposed 4-parallel radix-2\(^2\) feedforward architecture for the computation of the 16-point DIF FFT.
stage. This derivation of radix-2 from radix-2 can be observed in Figs. 1 and 2 for the particular case of N=16. Analogously, the radix-2^2 DIT FFT can be derived from the radix-2 DIT FFT. Contrary to DIF, for DIT the non-trivial rotations ϕ are moved to the previous stage instead of the following one.

### III. DESIGNING RADIX-2^2 FFT ARCHITECTURES:

The proposed architectures have been derived using the framework presented in. The design is based on analyzing the flowgraph of the FFT and extracting the properties of the algorithm. These properties are requirements that any hardware architecture that calculates the algorithm must fulfill. The properties of the radix-2^2 FFT are shown in Table I. The following paragraphs explain these properties and how they are obtained. The properties depend on the index of the data, l=b[s]=b[2], where (=) will be used throughout the paper to relate both the decimal and the binary representations of a number. This index is included in Fig. 2 both in decimal and in binary. On the one hand, the properties related to the butterfly indicate which samples must be operated together in the butterflies. This condition is b[s], both for DIF and DIT decompositions and means that at any stage of the FFT, butterflies operate in pairs of data whose indices differ only in bit b[s], where n= log_2 N is the number of stages of the FFT.

In Fig. 2 it can be observed that at the third stage s=3, data with indices l=12=1100 and l=14=1110 are processed together by a butterfly. These indices differ in bit b[3], which meets b[s]=b[2], since n= log_2 N= log_2 16=4 and, thus, b[s]= b[2]= b[3]= b[1]. On the other hand, there are two properties for rotations. At odd stages of the radix-2^2 DIF FFT only those samples whose index fulfills b[n]=b[n]=1 have to be rotated. These rotations are trivial and the symbol ( ) indicates the logic AND function. For the 16-point radix-2 FFT in Fig. 2 only samples with indices 12, 13, 14, and 15 must be rotated at the first stage. For these indices b[1], b[3]=1 is fulfilled, meeting the property b[n]=b[n]=1, since and . Conversely, at even stages rotations are non-trivial and they are calculated over indexed data for which b[n]+ b[n]=1 , where the symbol (+) indicates the logic OR function.

### IV. RADIX-2^2 FEEDFORWARD FFT ARCHITECTURES

This section presents the radix-2^2 feedforward architectures. First, a 16-point 4-parallel radix-2^2 feedforward FFT architecture is explained in depth in order to clarify the approach and show how to analyze the architectures. Then, radix-2^2 feedforward architectures for different numbers of parallel samples are presented. Fig. 3 shows a 16-point 4-parallel radix-2^2 feedforward FFT architecture. The architecture is made up of radix-2 butterflies (R2), non-trivial rotators, trivial rotators, which are diamond-shaped, and shuffling structures, which consist of buffers and multiplexers. The lengths of the buffers are indicated by a number.

The architecture processes four samples in parallel in a continuous flow. The order of the data at the different stages is shown at the bottom of the figure by their indices, together with the bits b[i] that correspond to these indices. In the horizontal, indexed samples arrive at the same terminal at different time instants, whereas samples in the vertical arrive at the same time at different terminals. Finally, samples flow from left to right. Thus, indexed samples (0, 8, 4, 12) arrive in parallel at the inputs of the circuit at the first clock cycle, whereas indexed samples (12, 13, 14, 15) arrive at consecutive clock cycles at the lower input terminal. Taking the previous considerations into account, the architecture can be analyzed as follows. First, it can be observed that butterflies always operate in pairs of samples whose indices differ in bit b[s], meeting the property in Table I. For instance, the pairs of data that arrive at the upper butterfly of the first stage are: (0, 8), (1, 9), (2, 10), and (3, 11). The binary representation of these pairs of numbers only differ in b[3]=b[4]=b[5]=1, so the condition is fulfilled. This property can also be checked for the rest of the butterflies in a similar way.

![Data Shuffling Circuit](image)

Second, Table I shows that rotations at odd stages are trivial and only affect samples whose indices fulfill b[n]=b[n]=1=1. By particularizing this condition for the first stage, b[3]=b[2]=b[1]=1 is obtained. In the architecture shown in Fig. 3 the indices that fulfill this condition are those of the lower edge and, thus, a trivial rotator is included at that edge. On the other hand, the condition for non-trivial rotations at even stages is b[n]=b[n]=b[2]=b[1]=1 being for the second stage. As for all indexed samples at the upper edge of the second stage, this edge does not need any rotator. Conversely, for the rest of edges b[3]+b[2]=1, so they include non-trivial rotators.

The rotation memories of the circuit store the coefficients of the flow graph. It can be seen that the coefficient associated to each index is the same as that in the flow graph.
of Fig. 2. For instance, at the flow graph the sample with index I=14 has to be rotated by Φ=6 at the second stage. In the architecture shown in Fig.3 the sample with index I=14 is the third one that arrives at the lower edge of the second stage. Thus, the third position of the rotation memory of the lower rotator stores the coefficient for the angle Φ=6.

Thirdly, the buffers and multiplexers carry out data shuffling. These circuits have already been used in previous pipelined FFT architectures [4], [17]–[20], and Fig. 4 shows how they work. For the first L clock cycles the multiplexers are set to “0″, L being the length of the buffers. Thus, the first samples from the upper path (set A) are stored in the output buffer and the first L samples from the lower path (set C) are stored in the input buffer. Next, the multiplexer changes to “1″, so set C passes to the output buffer and set D is stored in the input buffer. At the same time, sets A and B are provided in parallel at the output. When the multiplexer commutes again to “0″, sets C and D are provided in parallel. As a result, sets B and C are interchanged. Finally, the control of the circuit is very simple: As the multiplexers commute every L clock cycles and L is a power of two, the control signals of the multiplexers are directly obtained from the bits of a counter.

VI. CONCLUSION

This paper extends the use of radix-2^2 feedforward (MDC) FFT architectures. Indeed, it is shown that feedforward structures are more efficient than feedback ones when several samples in parallel must be processed. In this proposed paper we worked on the concept of Fast Fourier Transform (FFT) using both general existing architectures and proposed feedforward architecture and implemented. Here in this proposed work we mainly concentrate on DIF decomposition. Additionally, both DIF and DIT decompositions can be used. Finally, experimental results show that the designs are efficient both in area and performance than the existing general architectures, being possible to obtain throughputs of the order of G Samples/s as well as very low latencies.

VII. REFERENCES


