Pipelined Architectures for Real-Valued FFT and Hermitian-Symmetric IFFT With Real Datapaths

Sayed Ahmad Salehi, Rasoul Amirfattahi, and Keshab K. Parhi, *Fellow, IEEE*

***Abstract*—This brief presents novel parallel pipelined architectures for the computation of the fast Fourier transform (FFT) of real signals and inverse FFT of Hermitian-symmetric signals using only *real* datapaths. The real FFT structure is transformed by transferring twiddle factors to subsequent stages, such that each stage in the proposed flow graph contains one column of butterfly units and one column of twiddle factor blocks, and each column of the flow graph contains only N samples. This is a key requirement for the design of architectures that are based on only real datapaths. This structure is then mapped to pipelined architectures. The proposed architectures can be used with any FFT size or level of parallelism, which is a power of two. A systematic method to design architectures for FFTs with different levels of parallelism and radix values is presented. By modifying the FFT flow graph for real-valued samples, this methodology leads to architectures with fewer adders, delays, and interconnections.**

***Index Terms*—Fast Fourier transform (FFT), Hermitian-symmetric inverse FFT (IFFT), IFFT, parallel processing, pipelining, real datapath, real FFT (RFFT).**

I. Introduction

FAST Fourier Transform (FFT) and its inverse [inverse FFT (IFFT)] are important operations in digital signal processing applications [1]. Typically, FFT/IFFT operates over complex numbers and produces complex numbers. Many regular designs for computation of FFT and IFFT of complex numbers [complex FFT (CFFT)/complex IFFT (CIFFT)] have been presented [2]–[4]. There has been an increasing interest in the FFT/IFFT computation for real-valued samples [real FFT (RFFT)/real IFFT (RIFFT)], since many physical signals such as biomedical signals can be represented by real numbers. When the input samples are real in time domain, their frequency spectrum (RFFT output/RIFFT input) is Hermitian symmetric [1]; thus, approximately half of the computations are redundant. This property can be exploited to reduce the hardware complexity.

Many RFFT architecture designs and implementations have been presented in the literature [5]–[7]. First pipelined architectures, which are dedicated for RFFT, were proposed in [8] and [9]. Compared with other architectures that compute two N-point RFFT using an N-point CFFT architecture, these architectures are more efficient. Among them, the architecture in [8] is composed of only *real* datapaths. In this brief, the RFFT architectures were presented for four-parallel implementations using the radix-2 algorithm. Whether the approach in [8] can be generalized to arbitrary radix and whether two-parallel architectures can be used for RFFT using *only real datapaths* have remained open questions. Recently, pipelined architectures for RFFT have been proposed in [10], but these architectures are composed of *hybrid* datapaths consisting of both complex and real datapaths.

The contribution of this brief is twofold. First, it is shown that two-parallel RFFT architectures can be implemented using real datapaths only, as opposed to using hybrid datapaths, as in [10]. Second, a systematic method is presented to design RFFT/RIFFT architectures that can be implemented with any FFT size, any power-of-two radix, and any level of parallelism (including two-parallel architectures). All architectures presented in this brief use only *real* datapaths; this leads to savings in number of adders and delays and interconnect wires. Compared with the hybrid architectures in [10], the proposed architectures require only one type of butterfly (BF) compared with the four types used in [10]. Thus, the proposed designs are more regular. Although only decimation-in-frequency (DIF) architectures are presented in this brief, the proposed method can also be used to design architectures using decimation-in-time (DIT) algorithms.

The organization of the brief is as follows. Section II describes the RFFT algorithm. Section III presents the proposed method for the example of a 16-point two-parallel RFFT. Generalization of the method is presented in Section IV for N-point RFFT with different levels of parallelism and radix values. Section V provides hardware comparison and experimental results.

II. RFFT

The N-point discrete Fourier Transform (DFT) for a sequence $x(n)$ is defined as follows [1]:

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

where $W_N = e^{-j(2\pi/N)}$. By using the FFT algorithm, the N-point DFT is expressed in terms of a two-point DFT, where every two-point DFT is referred to as a BF.

It is easy to show that, for real-valued signals, we have

$$X[k] = X^*[N-k]. \tag{2}$$

Manuscript received April 1, 2013; accepted May 31, 2013. Date of publication July 11, 2013; date of current version August 10, 2013. This brief was recommended by Associate Editor C.-Y. Lee.

S. A. Salehi and R. Amirfattahi are with the Department of Electrical and Computer Engineering, Isfahan University of Technology, Isfahan 84156 83111, Iran (e-mail: saleh022@umn.edu; fattahi@cc.iut.ac.ir).

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

Color versions of one or more of the figures in this brief are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSII.2013.2268411

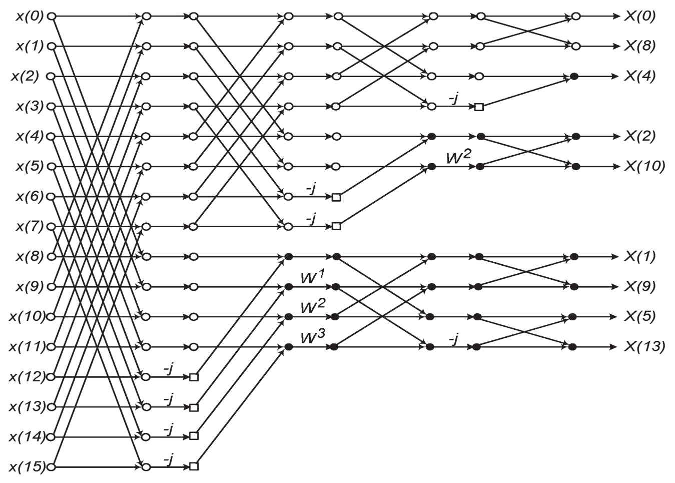
Fig. 1. FFT flow graph after removing redundant computations for RFFT.

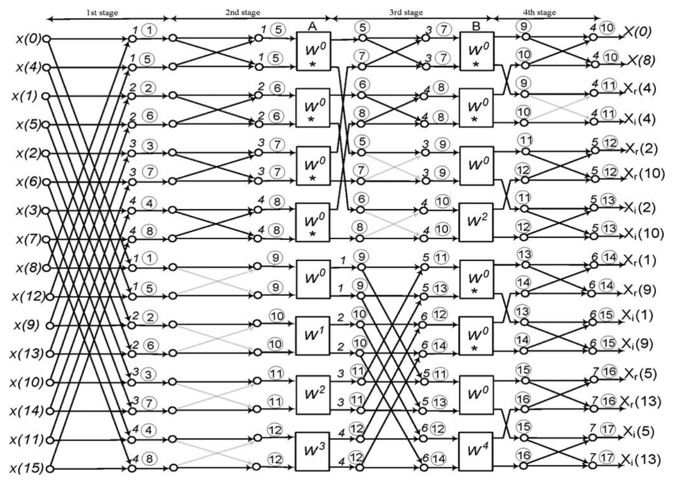
Fig. 2. Regular flow graph corresponding to Fig. 1. In this figure, real and imaginary parts have been separated.

Therefore, it is not necessary to compute all of the FFT coefficients. In [8], the computations of $(N/2-1)$ conjugate-symmetric samples were eliminated to obtain the FFT structure shown in Fig. 1. Before mapping it to a pipelined architecture, we need to transform it to a regular structure. In the next section, an algorithm for generating a regular structure and mapping it to a VLSI architecture is presented.

III. Proposed Algorithm and VLSI Architectures for RFFT Computation

Each real and imaginary datum, marked $\circ$ and $\square$, respectively, in Fig. 1 represents one real number, while each complex datum ($\bullet$) represents two real numbers. Based on that, every column in Fig. 1 contains N real numbers. Thus, by separating real and imaginary parts of complex data, the structure can be transformed to have N complete rows. Twiddle factors are transferred to subsequent stages as needed, such that the transformed flow graph contains one column of BFs followed by one column of W^k-blocks, and each column in the flow graph contains a total of N real and imaginary samples [8]. First and last stages do not need W^k-blocks. Fig. 2 shows the regular structure obtained after this transformation. The

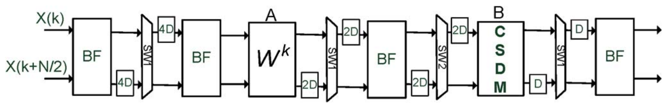
Fig. 3. Proposed two-parallel pipelined VLSI architecture for a 16-point DIF RFFT.

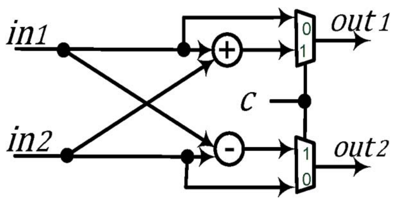
Fig. 4. Internal circuit of BFs in Fig. 3.

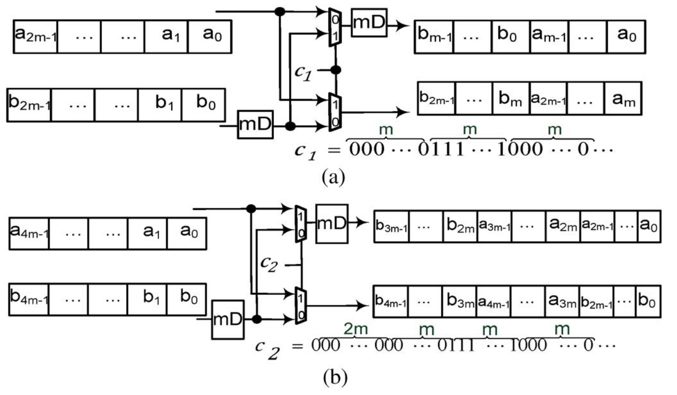
Fig. 5. (a) Dataflow for switch type SW1. (b) Dataflow for switch type SW2.

circled and italic numbers in Fig. 2 represent timing instances of the computations. The circled and italic timing instances are, respectively, used to derive two-parallel and four-parallel architectures in this brief. This structure is more suitable for the design of general parallel pipelined RFFT architectures than the structure in [8].

In the next step, each column of BFs and W^k-blocks is mapped to a BF and a W^k-block circuit, respectively. Fig. 3 shows the two-parallel pipelined architecture derived from the dataflow in Fig. 2. To the best of our knowledge, this is the first two-parallel pipelined architecture for RFFT computation using *real* datapaths. The internal circuit of the BF block in this architecture is shown in Fig. 4. For two real inputs, this BF adds or subtracts two real numbers, but for real–imaginary inputs, input data are simply transferred to the output. Control signal c controls switching between these two cases.

The multiplication by W^2 is the only W^k-block in the third stage that needs actual multiplication. It can be implemented by two addition operations and a scaling operation by the constant $1/\sqrt{2}$. The scaling operation can be implemented using canonical-signed-digit (CSD) multiplication [11]. Therefore, the column of W^k-blocks in the third stage is mapped to a CSD multiplier (CSDM), which is hardware efficient compared with a *complex* multiplier.

In order to synchronize data in accordance with the timing shown in Fig. 2, switches and delay elements, referred to as shuffling circuits, are used. Two types of switches are used in Fig. 3. Dataflow for switch type SW1 with $2m$ delay elements is shown in Fig. 5(a). Control signal c_1 for switch SW1 is periodic with period $2m$ clock cycles. For switch type SW2 with $2m$ delay elements, the first $2m$ samples of upper and

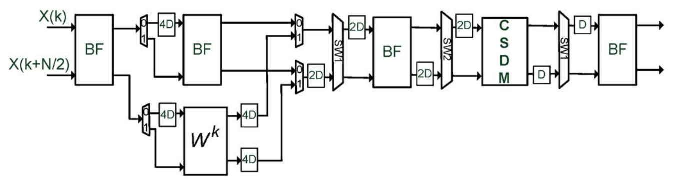

Fig. 6. Two-parallel pipelined architecture for 16-point radix-2 RFFT based on [8].

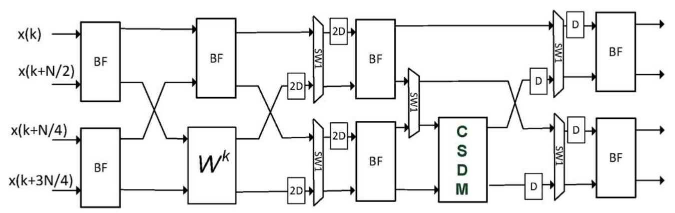

Fig. 7. Four-parallel pipelined architecture for 16-point DIF radix-2 RFFT.

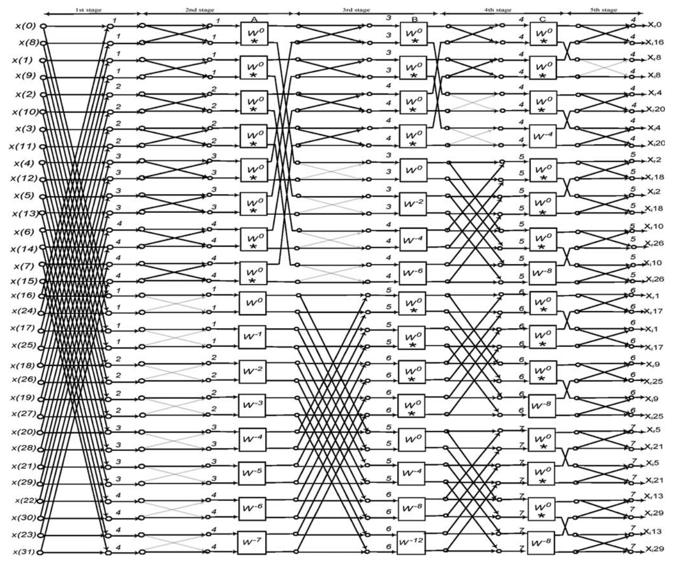

Fig. 8. Regular structure for 32-point radix-2 DIF RFFT.

lower inputs pass through the switch, respectively, to upper and lower outputs with m delays. After these $2m$ clock cycles, the switch operates in the same way as SW1. Fig. 5(b) illustrates the dataflow for switch type SW2. Control signal c_2 is initialized with $2m$ 0s followed by the same periodic pattern as c_1. For an N-point RFFT pipelined architecture, a single $(n-1)$-bit counter can be used to produce the required control signals for all of the switches in different stages, where $n = \log_2(N)$.

A two-parallel pipelined architecture for 16-point radix-2 RFFT using only real datapaths can be derived from the regular structure presented in [8]. Fig. 6 shows such an architecture that requires two demultiplexers. Compared with Fig. 6, our proposed architecture in Fig. 3 has two advantages: regular layout and fewer delay elements.

The proposed method for designing two-parallel architectures can be summarized using the following steps: 1) eliminating the computation of redundant outputs and transferring twiddle factors to subsequent stages, as described in [8], such that each column has N samples; 2) separating complex datapaths to *real* and *imaginary* datapaths; 3) obtaining regularity by transforming the flow graph to a structure with one BF column followed by one W^k-block column for each stage; and 4) constructing an appropriate schedule and mapping this regular structure to a pipelined architecture. Each BF column is mapped to one BF block, and each column of W^k-block is mapped to one W^k-block circuit.

IV. Generalization of the Algorithm

Here, the proposed algorithm is extended for: 1) higher levels of parallelization; 2) N-point RFFT; 3) radix-2^i algorithms; and 4) N-point RIFFT.

A. Level of Parallelism (L)

This section describes four- and eight-parallel pipelined architectures designed by using the proposed method. Architectures with higher level of parallelization can be designed using the folding technique described in [9] and [11].

To design an efficient pipelined architecture, we can simplify the regular structure before mapping it to a pipelined architecture. First, consider the design of a four-parallel architecture. In the second stage of the structure shown in Fig. 2, the four top W^k-blocks (marked by *) and four bottom BFs pass the data unaltered. Hence, these can be removed. Then, there are four BFs at the top of the second stage and four W^k-blocks at the bottom. Since a four-parallel architecture contains two hardware blocks for every stage, one BF block and one W^k-block can, respectively, calculate BF and W^k-block operations in the second stage. For the W^k-blocks in the third stage, we need only one W^k-block in the hardware architecture. This is because if two W^k-blocks are used, one of them will always pass the data unaltered. As mentioned earlier, the required W^k-block can be implemented by a CSDM. The structure after this simplification is mapped to the four-parallel architecture shown in Fig. 7. This architecture is very similar to the architecture proposed in [8].

Consider the design of an eight-parallel architecture for a 32-point DIF RFFT. Fig. 8 shows the regular structure achieved by the proposed method. The same simplification used in the four-parallel case can be used for the second, third, and fourth stages. For example, in the fourth stage, three W^k-blocks out of each four W^k-blocks pass through the input data without any calculation, and these can be removed. In a general case of an N-point RFFT, at stage $s \in [2, n-2]$, for each $N/2^{s-1}$ of W^k-blocks, the top $N/2^s$ ones can be removed. For $s = n-1$, three of each four W^k-blocks can be removed. For $N = 32$, W^k-blocks that can be removed are marked by * in Fig. 8. After this simplification, the structure in Fig. 8 is mapped to the eight-parallel architecture shown in Fig. 9. The italic numbers shown in Fig. 8 represent timing instances associated with the computation in Fig. 9. The third stage of the hardware architecture includes a three-input three-output switch SW3. In general, for N-point RFFT, one SW3 and one SW1 are required in stage $(n-2)$. For the first two clock cycles, inputs 1, 2, and 3 of SW3 are connected to outputs 1, 2, and 3, respectively. At the same time, top and bottom inputs of SW1 are connected to its top and bottom outputs, respectively. For all the other clock cycles, inputs 1, 2, and 3 of SW3 are connected to outputs 2, 3, and 1, respectively. At the same time, for SW1, top and bottom inputs are connected to bottom and top outputs, respectively.

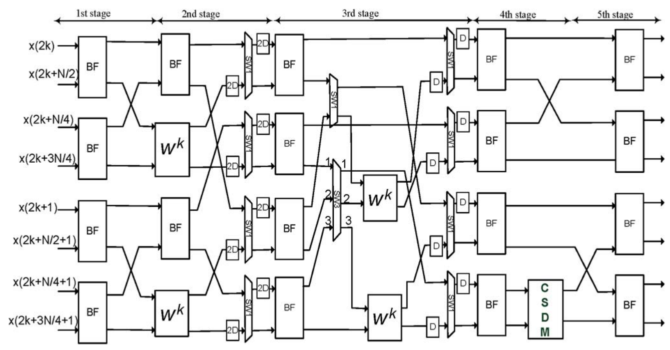
Fig. 9. Proposed eight-parallel pipelined architecture for 32-point DIF radix-2 RFFT.

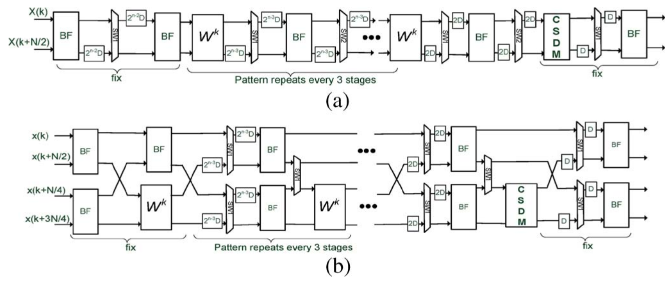
Fig. 10. General (a) two- and (b) four-parallel architectures for radix-2 N-point RFFT.

B. N-Point RFFT

Due to the regularity of the transformed structure, extension of the pipelined architectures for any N-point RFFT is straightforward. The general two- and four-parallel pipelined architectures for N-point RFFT can be implemented, as shown in Fig. 10. Similar architectures can be derived for eight and higher levels of parallelism.

The two-parallel architecture in Fig. 10(a) requires one CSDM and $(\log_2 N) - 3$ *complex* multipliers. Based on Fig. 10(a), the number of delay elements is calculated as

$$\begin{aligned} &2 + 4(2 + 2^2 + \cdots + 2^{n-3}) + 2 \times 2^{n-2} \\ &= 2 + 2^{n-1} + 4\sum_{i=1}^{n-3} 2^i \\ &= 2^{n-1} + 2^n - 6 = \frac{3N}{2} - 6. \end{aligned} \tag{3}$$

Similarly for a four-parallel architecture, $(\log_2 N) - 3$ complex multipliers are needed, and the number of required delay elements is given by

$$4(1 + 2 + 2^2 + \cdots + 2^{n-3}) = 4\left(\sum_{i=0}^{n-3} 2^i\right) = N - 4. \tag{4}$$

C. Higher Power-of-Two Radix-Based Architectures

Radix-2^i algorithms are used to decrease the number of stages with *complex* twiddle factors in order to decrease the number of required multipliers. The proposed method can be used to implement all power-of-two higher radix algorithms with reduced multiplication complexity.

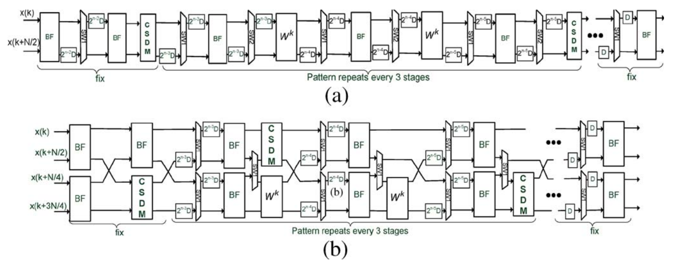
Fig. 11. General (a) two- and (b) four-parallel architectures for N-point radix-2^3 RFFT.

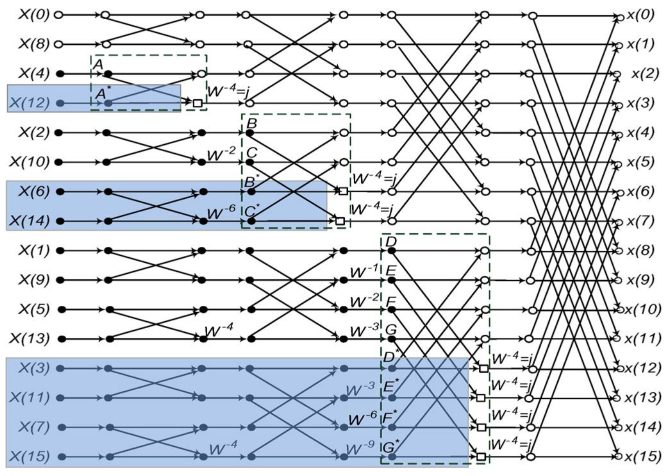
Fig. 12. Modified flow graph of 16-point DIF RIFFT.

For example, after transferring the twiddle factors in the flow graph of radix-2^3 FFT, it has two columns of *complex* W^k-blocks and one column of $W^{N/8}$-blocks per every three consecutive stages. This modified flow graph can be mapped to the pipelined architecture shown in Fig. 11. Here, the W^k-blocks in the second, fifth, eighth, ... stages are implemented by CSDM. This reduces area and power consumption. It may be noted that further twiddle factor transfers in the third, sixth, ninth, ... stages have been exploited beyond what would be necessary for satisfying real datapath requirement. This optimization step allows two complex multipliers in these stages to be replaced by a complex multiplier and a CSDM.

D. RIFFT

The N-point inverse DFT is defined as

$$x(n) = \frac{1}{N}\sum_{n=0}^{N-1} X[k]W_N^{-nk}, \qquad n = 0, 1, \ldots, N-1. \tag{5}$$

For RIFFT, some dedicated VLSI architectures have been proposed [6]; however, no pipelined architecture with *real* datapaths for RIFFT has been proposed yet. This section describes how the method presented in Section II can be used to derive pipelined architectures for RIFFT.

Fig. 12 shows the flow graph of a 16-point DIF RIFFT. Twiddle factors are transferred from the right of BFs inside the dashed boxes to their left to guarantee that every column contains N real or imaginary samples. The upper inputs of

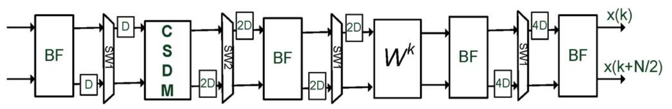

Fig. 13. Proposed two-parallel pipelined VLSI architecture for the 16-point DIF RIFFT.

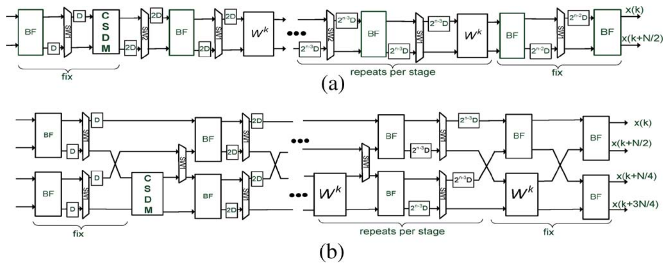

Fig. 14. General (a) two- and (b) four-parallel architectures for N-point RIFFT.

TABLE I
HARDWARE COMPLEXITY COMPARISON OF ARCHITECTURES FOR N-POINT RFFT

Pipelined architecture	Parallelization Level (L)	*Complex* Multiplier	Real adder	Delays
Radix-2 [8]	4	log_2N-3	$4log_2N$	N
Radix-2^3 [10]	2	log_8N-1	$2log_2N$	$3N/2-2$
	4	$2(log_8N-1)$	$4log_2N-2$	$7N/4-4$
Radix-2 Proposed	2	log_2N-3	$2log_2N$	$3N/2-6$
	4	log_2N-3	$4log_2N-2$	$N-4$
	8	$2log_2N-6$	$8log_2N-4$	$N-8$
Radix-2^3 Proposed	2	$2(log_8N-1)$	$2log_2N$	$3N/2-6$
	4	$2(log_8N-1)$	$4log_2N-2$	$N-4$
	8	$3(log_8N-1)$	$8log_2N-4$	$N-8$

these BFs (defined as $A, B, \ldots$) are complex conjugates of their lower inputs $(A^*, B^*, \ldots)$. Thus, the redundant operations represented by dark boxes can be deleted. Then, separating real and imaginary parts leads to a regular structure. The regular structure can be mapped to the two-parallel pipelined architecture shown in Fig. 13. Fig. 14 shows general two- and four-parallel N-point RIFFT pipelined architectures.

V. COMPARISON AND EXPERIMENTAL RESULTS

The hardware complexity of the proposed and prior pipelined RFFT architectures is summarized in Table I. Our proposed four-parallel architectures for radix-2 RFFT and architectures in [8] have similar hardware complexity because they are the same architectures achieved by different approaches. Compared with [10], for radix-2^3 RFFT algorithm, the proposed architectures require more multipliers for only the two-parallel architecture. However, the proposed architectures require same number of multipliers and less number of delays and interconnections when the level of parallelism is higher than two. One should note that our proposed architectures and the four-parallel architecture in [8] use only real datapaths, whereas the architectures in [10] make use of hybrid datapaths. Thus, our proposed architectures inherently require less number of interconnections.

TABLE II
EXPERIMENTAL RESULTS FOR THE PROPOSED L-PARALLEL N-POINT RFFT ARCHITECTURES

Architecture	L	N	slices	Freq.(MHz)	MS/s
Radix-2	2	16	612	461	922
		32	795	461	922
		64	994	459	918
	4	16	1139	438	1752
		32	1478	435	1740
		64	1849	435	1740
Radix-2^3	2	16	612	461	922
		32	707	461	922
		64	885	461	922
	4	16	1139	438	1752
		32	1275	437	1748
		64	1566	437	1748

Table II presents the synthesis results obtained for implementation of proposed architectures on a Xilinx Virtex-5 field-programmable gate array, i.e., XC5VLX110T. The numbers were represented using fixed-point 2's complement representation with 16-bit word length. In these implementations, none of the DSP48E slices and block random access memory devices is used; instead only distributed memory and lookup table slices are used. As expected, radix-2^3 requires the least number of hardware slices. Table II shows that, for larger N, the benefit of radix-2^3 is more significant.

ACKNOWLEDGMENT

The authors would like to thank the Associate Editor and the reviewers for their numerous constructive comments.

REFERENCES

[1] A. V. Oppenheim, R. W. Schafer, and J. R. Buck, *Discrete-Time Signal Processing*., 2nd ed. Englewood Cliffs, NJ, USA: Prentice-Hall, 1998.
[2] S. He and M. Torkelson, "Design and implementation of a 1024-point pipeline FFT processor," in *Proc. IEEE Custom Integr. Circuits Conf.*, Santa Clara, CA, USA, May 1998, pp. 131–134.
[3] Y. W. Lin, H. Y. Liu, and C. Y. Lee, "A 1-GS/s FFT/IFFT processor for UWB applications," *IEEE J. Solid-State Circuits*, vol. 40, no. 8, pp. 1726–1735, Aug. 2005.
[4] Y.-W. Lin and C.-Y. Lee, "Design of an FFT/IFFT processor for MIMO OFDM systems," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 54, no. 4, pp. 807–815, Apr. 2007.
[5] H. Sorensen, D. Jones, M. Heideman, and C. Burrus, "Real-valued fast Fourier transform algorithms," *IEEE Trans. Acoust., Speech Signal Process.*, vol. 35, no. 6, pp. 849–863, Jun 1987.
[6] H. F. Chi and Z. H. Lai, "A cost-effective memory-based real-valued FFT and Hermitian symmetric IFFT processor for DMT-based wire-line transmission systems," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2005, vol. 6, pp. 6006–6009.
[7] Y. Voronenko and M. Püschel, "Algebraic signal processing theory: Cooley-Tukey type algorithms for real DFTs," *IEEE Trans. Signal Process.*, vol. 57, no. 1, pp. 205–222, Jan. 2009.
[8] M. Garrido, K. K. Parhi, and J. Grajal, "A pipelined FFT architecture for real-valued signals," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 56, no. 12, pp. 2634–2643, Dec. 2009.
[9] M. Ayinala, M. Brown, and K. K. Parhi, "Pipelined parallel FFT architectures via folding transformation," *IEEE Trans. Very Large Scale Intergr. (VLSI) Syst.*, vol. 20, no. 6, pp. 1068–1081, Jun. 2012.
[10] M. Ayinala and K. K. Parhi, "FFT architectures for real-valued signals based on radix-2^3 and radix-2^4 algorithms," *IEEE Trans. Circuits Syst. I, Reg. Papers,* doi:10.1109/TCSI.2013.224625, to be published.
[11] K. K. Parhi, *VLSI Digital Signal Processing Systems: Design and Implementation*. Hoboken, NJ, USA: Wiley, 1999.