A Novel Memory-Based FFT Architecture for Real-Valued Signals Based on a Radix-2 Decimation-In-Frequency Algorithm

Zhen-Guo Ma, Xiao-Bo Yin, and Feng Yu, *Member, IEEE*

***Abstract*—This brief presents a novel architecture for memory-based fast Fourier transform (FFT) computation for real-valued signals based on radix-2 decimation-in-frequency algorithm. A superior strategy of stage partition for the real FFT (RFFT) is proposed to minimize the computation clock cycles and maximize the utilization of the processing element (PE). The PE employed in our RFFT architecture can process four inputs in parallel by using two radix-2 butterflies and only two multiplexers. The proposed memory-addressing scheme and control of the multiplexers can be expressed in terms of a counter according to the RFFT computation stage. Furthermore, the proposed RFFT architecture can support more PEs in two dimensions as well. Compared with prior works, the proposed RFFT processors have the advantages of fewer computation cycles and lower hardware usage. The experiment shows that the proposed processor reduces the computation cycles by a factor of 17.5% for a 32-point RFFT computation compared with a recently presented work while maintaining lower hardware usage and complexity in the PE design.**

***Index Terms*—Fast Fourier transform (FFT), memory-addressing scheme, memory based, real FFT (RFFT), real-valued signals.**

I. Introduction

FAST Fourier transform (FFT) is widely used in the field of digital signal processing, such as speech, audio, image, radar, and biomedical signal processing [1]. FFT generally operates over complex numbers, and much research has been done on the efficient architecture for the complex FFT (CFFT) [2]. Nowadays, the interest in the computation of FFT for real-valued signals is increasing since most of the physical signals are real [3]–[5]. However, not much research has been done on optimizing the architecture until recently for the FFT computation of real-valued signals.

The presented architecture for the CFFT computation cannot achieve the same efficiency for the real FFT (RFFT) computation because, when the input samples are real-valued signals, the spectrum of the FFT is symmetric and approximately half of the operations are redundant [6], [7]. Most RFFT architectures presented can be divided into two categories, i.e., pipelined [8] and memory-based architectures [9]. Taking the advantage of structural regularity in VLSI implementation, some pipelined RFFT architecture designs have been presented to achieve higher performance in computing time by employing many processing elements (PEs). The memory-based RFFT architectures employ one or several PEs to provide the tradeoff between hardware cost and speed performance in low- and moderate-speed applications, which are suited for large-size RFFT computation. These architectures are adopted in many applications such as optical coherence tomography in image processing [1], orthogonal frequency-division multiplexing and discrete multitone in communication [9], and wireless sensor network [10]. The focus of this brief is on memory-based architectures for RFFT computation, which are also referred to continuous-flow or in-place architectures. Some memory-based RFFT architectures have been proposed to achieve low hardware usage and high hardware resource utilization [10]–[12].

In this brief, we propose a novel memory-based architecture that computes the RFFT based on the modified radix-2 decimation-in-frequency (DIF) algorithm in [13]. The modified algorithm requires the lowest number of operations for radix-2 by computing only half of the output samples as presented. The modified algorithm separates the data into real and imaginary components with real data paths in a regular flow graph. Due to this, the word length of the required memory can be W instead of $2W$, where W is the word length chosen to represent either the real or imaginary component. An in-place FFT architecture based on the modified radix-2 algorithm [12] has been recently proposed to achieve the lower area-time (AT) product for RFFT, where area corresponds to the data-path area and time corresponds to the number of cycles. However, the RFFT architecture in [12] does not achieve the best hardware utilization. The key contribution of this brief is the design of a memory-based architecture based on a new strategy of RFFT stage partition that achieves lower AT, as compared with the prior works.

The organization of this brief is as follows. Section II first describes the radix-2 RFFT algorithm and then presents a new strategy of the RFFT stage partition to achieve fewer computation cycles. Then, the RFFT architecture and the parallelism exploitation in two dimensions are detailed in Section III. Finally, the proposed architecture is compared with prior designs and the implementation result is shown in Section IV.

II. New Strategy of RFFT Stage Partition

The N-point discrete Fourier transform for a sequence $x(n)$ is defined as

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk}, \quad 0 \le k \le N-1 \tag{1}$$

Manuscript received December 14, 2014; revised February 19, 2015; accepted April 13, 2015. Date of publication May 20, 2015; date of current version August 28, 2015. This brief was recommended by Associate Editor G. Masera.

The authors are with Zhejiang University, Hangzhou 310027, China (e-mail: 850501@zju.edu.cn; xiaobo@zju.edu.cn; osfengyu@zju.edu.cn).

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.2015.2435522

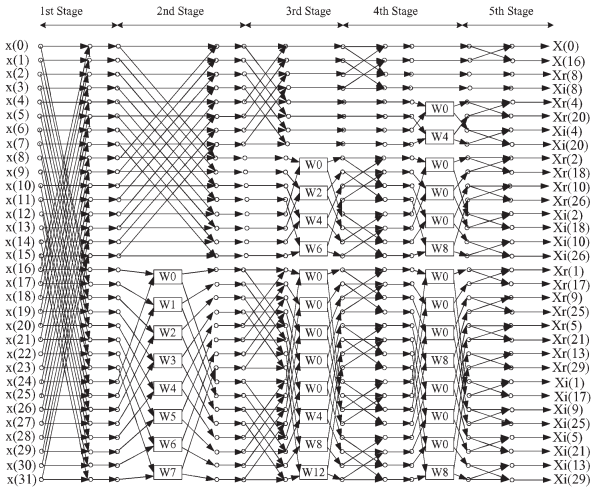

Fig. 1. Traditional strategy of stage partition for 32-point RFFT.

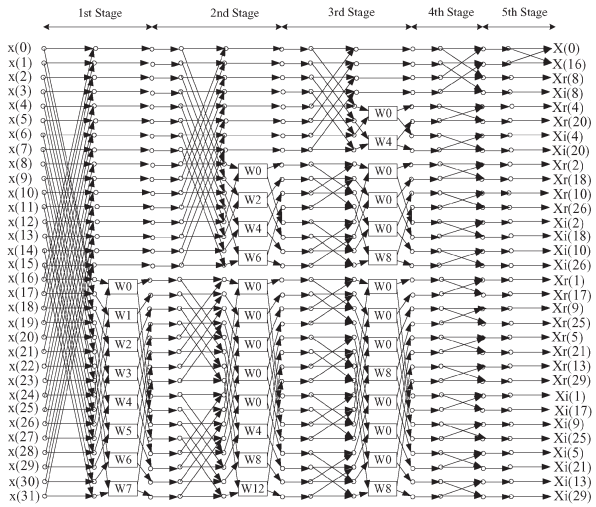

Fig. 2. Presented strategy of stage partition for 32-point RFFT.

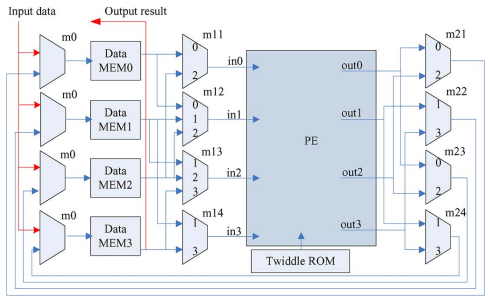

Fig. 3. Proposed high-level RFFT architecture with one PE.

where $W_N = e^{-j(2\pi/N)}$ is often referred to as the twiddle factor. When the samples are real, we have $X(k) = X^*[N-k]$. Therefore, it is not necessary to compute all of the FFT coefficients. A modified radix-2 flow graph [13] that contains only real data paths in a regular way is shown in Fig. 1. The N-point RFFT computation can be divided in n stages, where n is equal to the value of $\log_2 N$. As the entire data path is real, the word length of the required memory can be reduced to half of the RFFT computation of the complex samples. Furthermore, all the butterflies in the flow graph only process real samples, which require two real adders instead of two complex adders (CAs). The proposed architecture takes the advantage of the reduced numbers of the real FFT with respect to the CFFT.

The traditional strategy of stage partition for the 32-point RFFT computation is shown in Fig. 1 [8.13]. The RFFT computations of the first stage and the last stage do not have multiplication. There are $N/2$ complex additions in the first stage, whereas only $N/4$ complex additions are computed in the second stage. Based on the traditional strategy of stage partition, a new in-place RFFT architecture [12] has been recently proposed to achieve the lower AT product. For the processor with one PE, the number of the required computation cycles for N-point RFFT can be calculated with $N/4 * \log_2 N$. The in-place RFFT architecture does not achieve the full utilization of the PE resource because the complex multiplier (CM) in that PE is not processing during the first and last stages, and one CA is bypassed during the middle stages, particularly the second stage.

To further reduce the RFFT computation clock cycles, a new strategy of the stage partition is proposed, as shown in Fig. 2. The multiplication and addition both appear in all the stages of the RFFT, except the last two stages. The multiplication can be ignored when the twiddle factor is W_0 as W_0 is a real value of one. Then, the number of the complex multiplication and complex addition required in all the stages, except the last two stages, can be considered as $N/4$ and $N/2$, respectively. Thus, the PE employed in the proposed architecture contains one CM and two CAs that can process four inputs in parallel. Compared with the traditional strategy, there is only one real addition in the last stage and more multiplications and additions are processed in the first two stages. The required computational cycle for N-point RFFT is only $(N/4 * (\log_2 N - 1) + 1)$ in the proposed processor with one PE. It is obvious that the proposed RFFT architecture based on the new strategy of the RFFT stage partition achieves fewer computation cycles, as compared with the traditional one. The factor of the reduced computation cycles for N-point RFFT can be calculated using

$$\left(1 - \frac{N/4*(\log_2 N - 1)+1}{N/4*\log_2 N}\right) * 100\% = \frac{1-4/N}{\log_2 N} * 100\%. \quad (2)$$

When N is 32, the reduced factor reaches 17.5%.

III. Proposed Architecture

Fig. 3 shows the high-level architecture of the proposed RFFT processor with one PE. Four memory banks are used to store the samples and intermediate data in the processor, and each memory bank can store $N/4$ words of data length W. Thus, the total capability of the required memory is N^*W bit. The memory supports dual-port access, where the data can be read and written at the same clock cycle. When the input samples are written to the memory banks, the four multiplexer named "m0" will select the input data path. After all the samples are stored, the four multiplexer named "m0" will select another input data path for the intermediate results. The four multiplexers named "m11, m12, m13, and m14" decide the input sequence of the memory banks, and the four multiplexers named "m21, m22, m23, and m24" choose which memory banks the intermediate results are written to. The number in the multiplexers is according to the number of the memory bank. To achieve lower hardware usage and simple logic design, the output way of the result could share the same data path with the read data path. When the result data are being read out, the reordering work

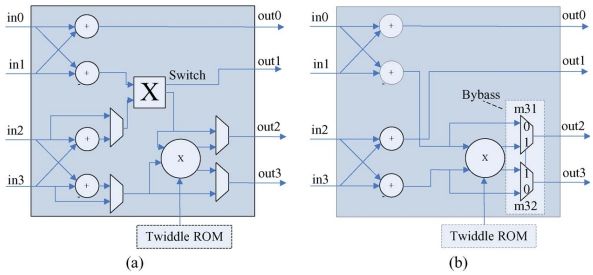

Fig. 4. Comparison of (a) the traditional PE and (b) the proposed PE for RFFT.

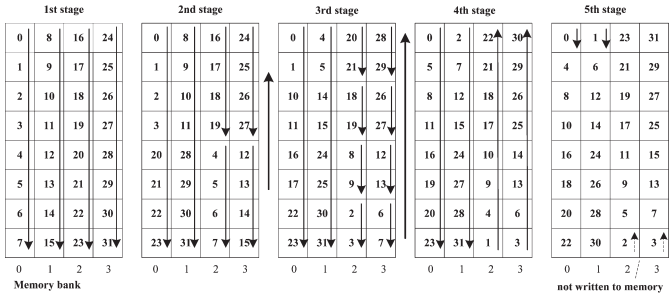

Fig. 5. Addressing scheme for 32-point RFFT with one PE.

could be done at the same time, and the next frame of N-point samples could be inputting simultaneously.

In the proposed RFFT architecture, a new PE that can process four samples in parallel is proposed by employing two butterflies and two multiplexers, as shown in Fig. 4(b). This choice of PE leads to less complexity and high throughout. The multiplexers named "m31 and m32" are used together to decide whether the multiplier is bypassed. There are four input ports named "in0, in1, in2, and in3" and four output ports named "out0, out1, out2, and out3" in the PE. Two real-valued data from some two memory banks and the twiddle factor from the bottom ROM are multiplied in the CM. The traditional PE that contains two butterflies and six multiplexers is shown in Fig. 4(a) [12], where the switch can be considered as two multiplexers. Compared with the traditional PE, the proposed PE achieves lower hardware usage and complexity.

A. Memory-Addressing Scheme

In this section, a new conflict-free memory-addressing scheme is detailed for the proposed RFFT processor with one PE. In the pipelined architecture [8], the intermediate data between the stages are stored in the registers, which act as first in first out. The number of required registers in the pipelined architecture decreases from the first stage to the last stage. In the proposed RFFT architecture, the intermediate data are written and read in groups. The size of the group decreases from the first stage to the last stage, which is similar to the way in the pipelined RFFT architecture.

An example of the addressing scheme for a 32-point RFFT computation is shown in Fig. 5, which shows how the data are stored in the memory banks at every stage. The first column corresponds to memory bank0, and the second column corresponds to memory bank1, and so on. The samples are stored in a natural order before the computation. After one computation of the PE, the four output data are written to the locations of the four input data. In this way, the memory bank conflicts are avoided during the computation in the next stage and our architecture achieves lowest hardware usage of memory. The arrows show how the data are read from and written into the memory banks during the stages. For example, the data with indices (0, 8, 16, and 24) are read out to the PE in the first stage, and the four result data are written in the same location. The data with indices (4, 12, 20, and 28) are read out, and the four result data are written in the interleaving location with indices (20, 28, 4, and 12), according to the data arrangement in the second stage.

TABLE I
ADDRESS PATTERNS FOR N-POINT RFFT COMPUTATION ($n = \log_2 N$)

Counter: W_{n-3}, W_{n-4},... W_0		Write/Read address
Memory bank 0,1	All stage	$W_{n-3}, W_{n-4}, ...W_0$
Memory bank 2,3	Stage 1 / Stage 2	$W_{n-3}, W_{n-4}, ...W_0$ / $\overline{W_{n-3}}, W_{n-4}, ...W_0$
	Stage 3 /	$\overline{W_{n-3}}, \overline{W_{n-4}}, ...W_0$ / .....
	Stage n-1 / Stage n	$\overline{W_{n-3}}, \overline{W_{n-4}}, ...\overline{W_0}$ / 0 (only one cycle)

As the PE can process four in parallel, each stage takes $N/4$ clock cycles for N-point RFFT computation. Thus, a $(n-2)$ bit counter $(W_{n-3}, W_{n-4}, \ldots W_0)$ can be used to count the clock cycles at each stage, where n is the stage number. The address of the memory banks can be expressed in terms of the counter at every stage, as shown in Table I. The read and write address patterns of memory bank0 and bank1 are the same for all stages, which can be represented as $W_{n-3}, W_{n-4} \ldots W_0$. The data are read and written in a serial order from address location 0 to $(N/4-1)$ at every stage for memory bank0 and bank1. The data in memory bank2 and bank3 are read and written in groups, which are different from bank0 and bank1. The size of the group goes down by a factor of 2 with every subsequent stage, as shown in Fig. 5. The addressing pattern for the second and third stages can be represented with $\overline{W_{n-3}}, W_{n-4}, \ldots W_0$ and $\overline{W_{n-3}}$, $\overline{W_{n-4}}, \ldots W_0$, respectively. Therefore, the read and write addresses of all the memory banks can be generated from only a counter by simply selecting the value or inverse value of each bit.

B. Control of Multiplexers for Memory Bank Assignment

There are two multiplexers in the PE and a total of 14 multiplexers in the proposed RFFT processor with one PE. Only when the input samples are written to the memory banks, the four multiplexer named "m0" selects the input data path. The four multiplexers named "m11, m12, m13, and m14" decide the input sequence of the memory banks, where the data are read out to the four input ports of the PE. For example, the multiplex named "m11" can select the input data from either memory bank0 or bank2, and the multiplex "m13" can select the input data from any memory of banks 1, 2, and 3, as shown in Fig. 3.

The control of the four multiplexers "m11, m12, m13, and m14" can be expressed with a counter at every stage, as shown in Table II. In the first stage, the four multiplexers make the data in memory bank 0,2,1,3 be read out to the "in0, in1, in2, and in3" input ports of PE, respectively. When the value of W_{n-3} is zero in the second stage, the data in memory bank 0,1,2,3 are read out to the "in0, in1, in2, and in3" input ports of PE, respectively. From the third to the $(n-1)$th stage, there are three bank select patterns according the value of the counter in different stages.

TABLE II
CONTROL OF THE MULTIPLEXERS DEPENDING ON THE VALUE OF THE COUNTER IN DIFFERENT STAGES

Multiplexer		m21,22,23,24		m11,12,13,14		m31,32
Stage	Part of the counter	Value type	Writebank seq.	Value type	Readbank seq.	multiplier is bypassed?
1				any	0,2,1,3	No
2	W_{n-3}	odd	2,3,0,1	zero	0,1,2,3	No
		else	0,1,2,3	else	2,0,3,1	
3	W_{n-3}, W_{n-4}	odd	2,3,0,1	zero	0,1,2,3	No
		else	0,1,2,3	odd	2,0,3,1	
				else	0,2,1,3	
.....		odd	2,3,0,1	zero	0,1,2,3	No
		else	0,1,2,3	odd	2,0,3,1	
				else	0,2,1,3	
n-1	W_{n-3}, W_{n-4},...W_0	odd	2,3,0,1	zero	0,1,2,3	Yes
		else	0,1,2,3	odd	2,0,3,1	
				else	0,2,1,3	
n		any	0,1,2,3	zero	0,1,2,3	Yes

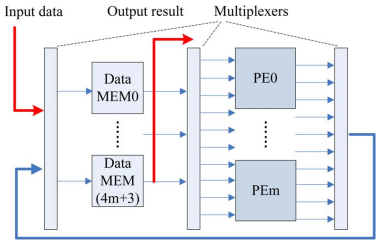

Fig. 6. Proposed high-level RFFT architecture with m PEs.

The bit width of the according counter increases as the stage number increases, as shown in Table II. Since there is only one real addition that is needed in the last stage, the result of the bottom CA is deserted. At the last stage, the data of memory bank 0,1,2,3 are read out to the "in0, in1, in2, and in3" input ports of PE, respectively, for only one clock cycle. After one addition in the last stage, the result can be unloaded to the output data path, and the next frame data can be unloaded for the N-point RFFT computation simultaneously.

The four multiplexers named "m21, m22, m23, and m24" select which memory banks the intermediate result from the output ports are written to. When the value of W_{n-3} is odd, the intermediate result of the first stage from "out0, out1, out2, and out3" ports is written to memory bank 2,3,0,1, respectively. When the value of W_{n-3} is not odd, the result of the first stage is written to memory bank 0,1,2,3, respectively. The two multiplexers "m31 and m32" in the PE decide whether the CM is bypassed. In Fig. 3, we can see that the computations of the last two stages do not contain any multiplication, and the two multiplexers "m31 and m32" make the multiplier bypassed.

C. Exploit Parallelism With Multiple PEs

The parallelism of the proposed RFFT architecture can be exploited in two dimensions: pipelined architecture (parallelism in multiple stages) [14] and parallel architecture (parallelism within a stage) [13]. Fig. 6 shows the high-level RFFT architecture with m PEs, which requires $4m$ memory banks.

In these pipelined RFFT architectures, the computation of each stage is implemented by one PE and a group of four memory banks. The number of multiplexers would proportionately increase as the number of the PEs increases. If the number of the employed PE is m, m memory groups of four banks will be required, each bank of which is required to store $N/4$ words at least. An example of the pipelined memory-based FFT architecture with four PEs is presented in [14], achieving lower usage of the memory resource that could be used for reference, in which the memory bank conflicts are well avoided. In these pipelined architectures, the number of the PEs could be a suitable value between one and the number of the stages.

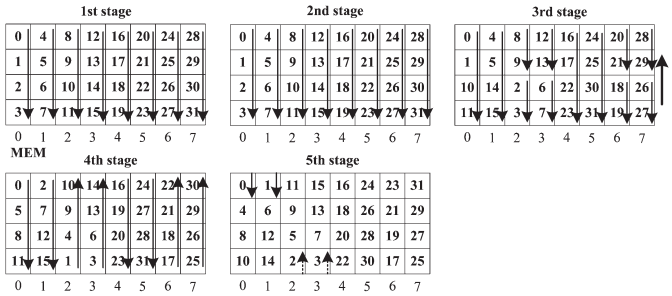

Fig. 7. Addressing scheme for 32-point RFFT processor with two PEs.

The parallelism of our proposed RFFT architecture can be also exploited within a stage. In the parallel RFFT architectures with m PEs, each memory bank needs to store $N/(4m)$ words of data length W at least, and the total required memory could be the same as N^*W bit. The resource of the required multiplexers would proportionately increase, and the design of the control unit will be more complex as the number of the PEs increases. To achieve the lowest usage of the memory resource, the memory bank conflicts should be also avoided during the computation. Compared with the pipelined architecture, the parallel architecture has the advantages of lower resource usage of memory. The limitation is that the number of the PEs (m) in the parallel architecture should be a power of 2. Otherwise, more memory resource would be used and the design of the control unit would become much more complex.

A new addressing scheme for 32-point RFFT computation for the processor with two PEs is proposed to achieve the lowest usage of the memory resource. At every stage of the RFFT computation, each memory bank in the proposed architecture with one PE shown in Fig. 5 can be regarded as the combination of some two memory banks in the architecture with two PEs shown in Fig. 7. It could be concluded that each memory bank in the proposed architecture with one PE can be regarded as the combination of some m memory banks in that parallel RFFT architecture with m PEs for all the stages. In this way, the proposed addressing scheme could be extended from the architecture with one PE to the parallel architectures with more PEs.

IV. COMPARISON AND EXPERIMENTAL RESULTS

In this section, we compare the hardware complexity and computation cycles with some proposed memory-based RFFT processors and a pipelined RFFT processor, as shown in Table III. A cost-effective memory-based RFFT processer based on the radix-4 algorithm is proposed in [11], which requires a complex radix-4 butterfly consisting of 12 CAs and 3 CMs. To further reduce the usage of the computing hardware resource, an in-place RFFT processor has been recently proposed in [12], achieving the same the computation cycles as that in [11]. The PE in [12] requires one CM, two CAs, and six multiplexers. Compared with [12], our proposed architecture achieves lower hardware usage and hardware complexity by using only two multiplexers in the PE. The proposed architecture is based on

TABLE III
COMPARISON OF THE RFFT PROCESSORS

	Proposed(1 PE)		Proposed(2 PE)		[12]		[11]		[9]		[9]		[15]		[13]	
Radix	Radix-2		Radix-2		Radix-2		Radix-4		Radix-2		Radix-4		Radix-2/4		Radix-2	
CA	2		4		2		12		2		8		8		$4\log_4 N$	
CM	1		2		1		3		1		3		3		$2(\log_4 N-1)$	
Memory	N/4 * W		N/8 * W		N/4 * W		N/8 * 2W		N/4 * 2W		N/4 * 2W		N/4 * 2W		N/4 * W	
banks	4		8		4		4		4		4		4		4	
N	#Cycle	AT	#Cycle	AT	#Cycle	AT	#Cycle	AT	#Cycle	AT	#Cycle	AT	#Cycle	AT	#Cycle	AT
256	449	5388	225	5400	512	6144	512	21504	1024	12288	256	9728	256	9728	64	4864
512	1025	12300	513	12312	1152	13824	768	32256	2304	27648	576	21888	640	24320	128	11264
1024	2305	27660	1153	27672	2560	30720	2560	107520	5120	61440	1280	48640	1280	48640	256	25600
2048	5121	61452	2561	61464	5632	67584	3854	161868	11264	135168	2816	107008	3072	116736	512	57344
4096	11265	135180	5633	135192	12288	147456	12288	516096	24576	294912	6144	233472	6144	233472	1024	126976

the new strategy of the RFFT stage partition, which further reduces the computation clock cycles. For N-point RFFT computation, the required computation cycles in [12] with one PE are $N/4 * \log_2 N$, whereas only $N/4 * (\log_2 N - 1) + 1$ computation cycles are required in our RFFT processor with one PE. When N is 32, the proposed RFFT architecture can reduce the computation cycles by a factor of 17.5% compared with that in [12]. We calculate the AT product by multiplying computation cycles with CA area as the unit (it is fair to assume that the area of a CM is ten times the area of a CA). Compared with the other memory-based architectures, the novel RFFT architecture achieves fewer computation cycles and better utilization of the PEs. From Table III, it can be concluded that the proposed design achieves the best performance in terms of AT product of memory-based architectures. Furthermore, the parallelism of the proposed RFFT architecture can be exploited in multiple stages and within a stage. The total computation clock cycles decrease approximately linearly with the increase in the number of PEs.

The number of PEs in the pipelined RFFT processor [13] increases with the size of the RFFT, which can process four in parallel. Since the design of the PEs, multiplexers, and control units of each stage in the pipelined processor can be various among the RFFT stages, its hardware optimization could be well done. Thus, the pipelined processor achieves high hardware utilization in the terms of AT product. The gap ratio (GR) of the AT product between the proposed architecture and the pipelined architecture can be calculated with the following equation:

$$GR = (AT\text{proposed} - AT\text{pipelined})/AT\text{pipelined} * 100\%. \quad (3)$$

The AT product of the proposed processor with one PE gets close to that of the pipelined processor, as the GR decreases from 10.8% to 6.5% when the size of FFT (N) increases from 256 to 4096. Compared with the pipelined architecture [13], the proposed architectures achieve lower hardware cost with one or several PEs, which would be more suitable for large-point RFFT computation in low- and moderate-speed applications.

Moreover, one obvious advantage of the proposed RFFT architecture is that the capability of the required memory can be reduced by a factor of 2, as compared with the traditional memory-based complex FFT processors in [9] and [15].

Table IV presents the synthesis results obtained for implementation of the proposed processors with ISE14.5 tool on a Xilinx Virtex-7 field-programmable gate array, i.e., XC7VX485T. The floating-point samples and twiddle factors are stored in the Block RAMs, each of which can store maximum data of 1000 words. The DSP48E are used for the implementation of floating-point multipliers and adders. As expected, the hardware cost in the processor with one PE is about half of that in the processor with two PEs.

TABLE IV
EXPERIMENTAL RESULT FOR THE PROPOSED N-POINT PROCESSORS

N	PE	Slice LUT	Slice Register	DSP48E	RAM*	Freq.(MHz)
1024	1	2840	2981	24	5	423.1
4096	1	2863	2992	24	8	410.2
1024	2	5545	5907	48	10	414.5
4096	2	5580	5937	48	16	409.6

*The Block RAMs includes both for the data and the twiddle factor.

ACKNOWLEDGMENT

The authors would like to thank the associate editor and all the anonymous reviewers for numerous constructive comments.

REFERENCES

[1] S.-N. Tang, F.-C. Jan, H.-W. Cheng, C.-K. Lin, and G.-Z. Wu, "Multi-mode memory-based FFT processor for wireless display FD-OCT medical systems," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 61, no. 12, pp. 3394–3406, Dec. 2014.

[2] H. F. Luo, Y. J. Liu, and M.-D. Shieh, "Efficient memory-addressing algorithms for FFT processor design," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, 2014, to be published. [Online]. Available: http://ieeexplore.ieee.org

[3] B. R. Sekhar and K. M. M. Prabhu, "Radix-2 decimation-in-frequency algorithm for the computation of the real-valued FFT," *IEEE Trans. Signal Process.*, vol. 47, no. 4, pp. 1181–1184, Apr. 1999.

[4] M. Ayinala and K. K. Parhi, "Parallel-pipelined radix-2^2 FFT architecture for real-valued signals," in *Conf. Rec. 44th ASILOMAR Signals, Syst. Comput.*, 2010, pp. 1274–1278.

[5] 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*, vol. 60, no. 9, pp. 2422–2430, Sep. 2013.

[6] H. V. Sorensen, D. L. Jones, M. Heideman, and C. S. Burrus, "Real-valued fast Fourier transform algorithms," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-35, no. 6, pp. 849–863, Jun. 1987.

[7] H. Murakami, "Real-valued decimation-in-time and decimation-in-frequency algorithms," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 41, no. 12, pp. 808–816, Dec. 1994.

[8] S. A. Salehi, R. Amirfattahi, and K. K. Parhi, "Pipelined architectures for real-valued FFT and Hermitian-symmetric IFFT with real datapaths," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 60, no. 8, pp. 507–511, Aug. 2013.

[9] P.-Y. Tsai and C.-Y. Lin, "A generalized conflict-free memory addressing scheme for continuous-flow parallel-processing FFT processors with rescheduling," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 19, no. 12, pp. 2290–2302, Dec. 2011.

[10] A. Wang and A. P. Chandrakasan, "Energy-aware architectures for a real-valued FFT implementation," in *Proc. ISLPED*, 2003, pp. 360–365.

[11] 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 ISCAS*, 2005, pp. 6006–6009.

[12] M. Ayinala, L. Yingjie, and K. K. Parhi, "An in-place FFT architecture for real-valued signals," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 60, no. 10, pp. 652–656, Oct. 2013.

[13] 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.

[14] Z.-G. Ma, F. Yu, R.-F. Ge, and Z.-K. Wang, "An efficient radix-2 fast Fourier transform processor with ganged butterfly engines on field programmable gate arrays," *J. Zhejiang Univ.*, vol. 12, no. 4, pp. 323–329, 2011.

[15] B. G. Jo and M. H. Sunwoo, "New continuous-flow mixed-radix (CFMR) FFT processor using novel in-place strategy," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 52, no. 5, pp. 911–919, May 2005.