Continuous-flow Parallel Bit-Reversal Circuit for MDF and MDC FFT Architectures

Sau-Gee Chen, Shen-Jui Huang, Mario Garrido Gálvez and Shyh-Jye Jou

Sau-Gee Chen, Shen-Jui Huang, Mario Garrido Gálvez and Shyh-Jye Jou, Continuous-flow Parallel Bit-Reversal Circuit for MDF and MDC FFT Architectures, 2014, IEEE Transactions on Circuits and Systems Part 1: Regular Papers, (61), 10, 2869-2877.
http://dx.doi.org/10.1109/TCSI.2014.2327271

Continuous-flow Parallel Bit-Reversal Circuit for MDF and MDC FFT Architectures

Sau-Gee Chen, Shen-Jui Huang, Mario Garrido, *Member IEEE*, and

Shyh-Jye Jou, Senior *Member*, *IEEE*

***Abstract*—This paper presents a bit reversal circuit for continuous-flow parallel pipelined FFT processors. In addition to two flexible commutators, the circuit consists of two memory groups, where each group has *P* memory banks. For the consideration of achieving both low delay time and area complexity, a novel write/read scheduling mechanism is devised, so that FFT outputs can be stored in those memory banks in an optimized way. The proposed scheduling mechanism can write the current successively generated FFT output data samples to the locations without any delay right after they are successively released by the previous symbol. Therefore, total memory space of only *N* data samples is enough for continuous-flow FFT operations. Since read operation is not overlapped with write operation during the entire period, only single-port memory is required, which leads to great area reduction. The proposed bit-reversal circuit architecture can generate natural-order FFT output and support variable power-of-2 FFT lengths.**

***Index Terms*—fast Fourier transform (FFT), natural-order FFT output, bit-reversal circuit, MDF, MDC**

I. Introduction

FAST FOURIER TRANSFORM (FFT) is widely used in various signal processing applications, such as spectrum analysis, image and video signal processing, and communication systems. Over the past decades, various FFT hardware architectures have been investigated, including pipelined FFT architectures and memory-based FFT architectures. Pipelined FFTs include single-path delay feedback (SDF) [1-2], single-path delay commutator (SDC) [3-5], multi-path delay feedback (MDF) [6-8], and multi-path delay commutator (MDC) [9-12] architectures. They have the advantage of high throughput, but demand high area cost especially for long-length FFTs. In contrast, memory-based FFT architectures usually have low area cost, because smaller numbers of butterfly processing elements (PE) are adopted to sequentially execute all the butterfly operations. Accordingly, their throughputs are often limited.

Recently, parallel pipelined FFT architectures [6-13] were proposed to enhance throughput by increasing parallelism of the whole architecture. As such, they can meet the demand of extremely high data rates of current state-of-art wireless communication systems, such as UWB (Ultra Wideband), IEEE 802.15.3c, or IEEE 802.11ac/ad. Two major function blocks should be designed for pipelined FFT processors, one is the FFT architecture itself and the other one is the bit-reversal circuit. The function of the bit-reversal circuit is to convert the non-natural output order of the FFT architecture to natural order. This feature is especially important for communication systems, because FFT processors are usually followed by frequency-domain equalizer which requires timely and natural-order input data. However, much fewer works are dedicated to bit-reversal circuit design in the literature until recent years, compared to the amount of works on FFT architecture designs. For general memory-based FFT architectures, there are memory addressing schemes [14-16], which facilitate natural-order FFT outputs. For pipelined FFT, bit-reversal circuits must support continuous-flow processing for the consideration of seamless generation of FFT outputs, due to contiguous inputs. Several works in the literature [2-5], [17-19], proposed bit-reversal circuits for single-path pipelined FFT architectures. For parallel pipelined FFTs, the design of the reordering circuits is even more challenging as it requires to reorder multiple concurrent FFT outputs simultaneously. Thus, only a few works in the literature discuss this problem [9-10], [12]. Among them, reordering circuits for parallel data are described in [9-10]. The circuit proposed in [9] calculates the bit reversal for parallel output data, but its hardware complexity is high. On the other hand, the outputs of FFTs in [10] are in an order different from bit reversal, and therefore the reordering circuit is only applicable to this specific order.

This work proposes a new bit-reversal circuit for parallel data that can be used for both MDC and MDF FFT architectures. The main contributions of this work are twofold. First, it is the first parallel bit-reversal circuit based on single-port memory. Besides, it is area-efficient, as the total memory size is *N*, where *N* is FFT length. Second, the proposed reordering mechanism is regular and flexible for supporting general power-of-2 FFT sizes, as well as variable-length bit reversal. The rest of this

Manuscript received March 17, 2014. This work is supported in part by National Science Council, Taiwan under the grants of NSC 101-2220-E-009 -025 and NSC 101-2219-E-009 -020.

S.G. Chen and Shyh-Jye Jou are with the Dept. of Electronics Engineering, National Chiao Tung University, Hsinchu 300, Taiwan, R.O.C. (e-mails: sgchen@cc.nctu.edu.tw; jerryjou@mail.nctu.edu.tw).

Shen-Jui Huang is with Novatek Corp., Hsinchu 300, Taiwan, R.O.C. (e-mail: shenray_huang@g2.nctu.edu.tw).

Mario Garrido is with the Dept. of Electrical Engineering, Linköping University, SE-581 83 Linköping , Sweden. (e-mail: mariog@isy.liu.se).

article is organized as follows. In Section II, existing bit-reversal circuits are reviewed. In Section III, the design problem for a parallel bit reversal circuit is formulated. In Section IV, the proposed bit-reversal circuit is presented. Implementations and comparisons with existing bit reversal circuits are made in Section V, followed by conclusions in Section VI.

II. Review of Existing Bit Reversal Circuits

There are various bit-reversal addressing schemes proposed in the literature. For non-continuous data flow, the schemes proposed in [20-23] focus on calculating the bit reversal on data stored in a memory. In [24-25], address generators for memory-based FFTs are proposed. Finally, for continuous data flow, solutions to bit reversal on serial data were provided in [2-5], [11], [17-19], and solutions for parallel data are provided in [9-10], [12].

A. Bit-reversal circuit for single-path serial data

In [17], the bit reversal on serial data is calculated using a double buffering strategy. This consists of two memories of size N where even and odd FFT output sequences are written alternatively in the memories. The bit reversal can also be calculated using a single memory of size N. This is achieved by generating the memory address in natural and bit-reversed order, alternatively for even and odd sequences [18]. The bit reversal circuit in [11] targets real-valued FFTs. Although the architectures in [11] are for parallel data, the bit reversal circuit only applies to serial data. For SDC FFT architectures, the output reordering can be calculated by using two memories of $N/2$ addresses [3-5]. Alternatively, the output reordering circuit can be integrated with the last stage of the FFT architecture [3-5]. Finally, in [19], a novel circuit for calculating bit reversal on serial data is proposed. The circuit consists of cascaded buffers and multiplexers, which can flexibly convert the bit-reversed output for common FFT radices, including radix-2, radix-2^k, radix-4, and radix-8. This approach provides the optimum circuits for bit reversal on serial data with minimum memory space.

B. Bit-reversal circuits for parallel data

For parallel pipelined FFTs, only few works in the literature propose solutions to reorder the output data in parallel FFT architectures [9-10], [12]. In [9], a bit-reversal circuit for 8-parallel data is proposed. For an N-point FFT, this circuit requires an N-address memory for each parallel stream. In [10], the outputs of the FFT are provided in an order different to bit-reversal. Thus, its reordering circuit is specific for the FFT architecture it proposed, but not for other MDC and MDF FFT architectures. Finally, [12] presents parallel radix-2^k MDC FFT architectures. It also discusses the possibility of reordering the bit-reversed outputs by using a total memory of N-(N/P). However, as the paper focuses on the FFT architectures, the bit reversal circuit is not described.

III. Problem Formulation of Parallel Bit-Reversal Circuit

Given an N-point discrete Fourier transform (DFT):

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn}, \quad k = 0, ..., N-1. \qquad (1)$$

where $x(n)$ and $X(k)$ denote the input and output of the DFT respectively, and $W_N^{kn} \equiv e^{-j2\pi kn/N}$, which is called twiddle factor. For efficient implementation of FFT operations, radix-2^k FFT algorithms are often applied. Besides, parallel pipelined architectures are often adopted to realize the radix-2^k FFT algorithms [6-8], because they can offer higher throughput than SDF or SDC pipelined architectures. As shown in Fig. 1, a pipelined FFT processor accepts P-parallel natural-order FFT input, and generates P-parallel bit-reversed FFT output, where P is the parallelism and $BR(k)$ is the bit-reverse representation of index k. First, to convert parallel FFT output to natural-order FFT output, a memory group partitioned into P memory banks is required. Denote the m-bit binary representation of k as $k_{m-1}k_{m-2}\ldots k_0$, where $m = \log_2 N$, the bit-reversed representation of k is shown in Fig. 2. In the figure, the q LSB bits represent the path index, and m-q MSB bits denote time index t, where $q = \log_2 P$, and $t \in \{0, 1, \ldots, (N/P)-1\}$, which denotes the time index. Since each set of P adjacent $X(k)$s (i.e., $\{X(Pt), X(Pt+1), \ldots, X(Pt+P-1)\}$) differ only in bits $\{k_0, k_1, \ldots, k_{q-1}\}$, their output path indices are the same. Therefore, they will be saved to the same memory bank if FFT outputs from P paths (i.e., path 0 ~ path P-1) are directly written to the corresponding P memory banks (i.e., bank 0 ~ bank P-1). This implies that it is impossible to provide P-parallel natural-order outputs to the next-stage functional block due to conflicting memory accesses. Therefore, to avoid memory conflict, a suitable reordering mechanism should be designed so that output from each path can be switched to proper memory bank. Second, considering continuous-flow FFT operation, generally two groups of memory are required for the purpose of acting as ping-pong buffers during each FFT output period (of N/P clock cycles). However, such architecture has the drawback of inefficient memory utilization, because memory space released after each readout cannot be immediately accessed during that output period. Thus, the problem of calculating the bit reversal of the FFT outputs translates into finding an efficient strategy to access the P memory banks.

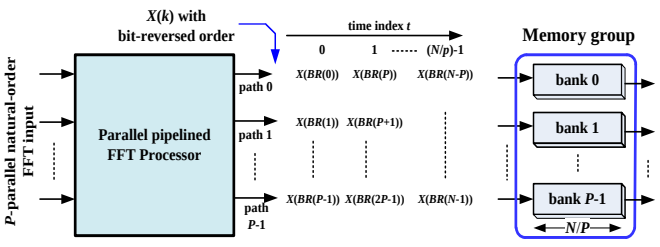

Fig. 1. Bit-reversed output ordering of parallel pipelined FFT processors.

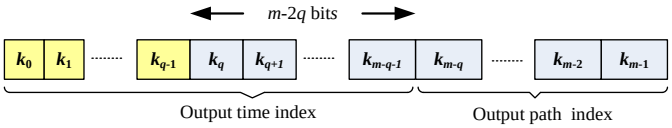

Fig. 2. Bit-reversed representation of k.

IV. Proposed Parallel Bit Reversal Circuit

Based on previous discussion, a new parallel bit reversal circuit for parallel pipelined FFT processors is proposed. As shown in Fig. 3, the architecture supports continuous-flow operation and calculates the bit reversals on P parallel inputs. The architecture is composed of input and output commutators, two groups of memory banks, and one controller. The Write Commutator, denoted as CMT_WR, plays the role of switching P FFT processor outputs to proper memory banks according to a pre-defined switching mechanism, which will be explained later. The Read Commutator, denoted as CMT_RD, helps to switch the P memory banks' output to proper output paths. The memory is partitioned into two single-port memory groups, A and B, each containing P memory banks. Furthermore, each memory bank stores $N/(2P)$ data samples, leading to a total memory size N. Between the memory and the Read Commutator, multiplexers are used to select the memory groups. Finally, the control block generates the memory addresses for read/write operations in each clock cycle. In addition, it also generates the control signals for commutators.

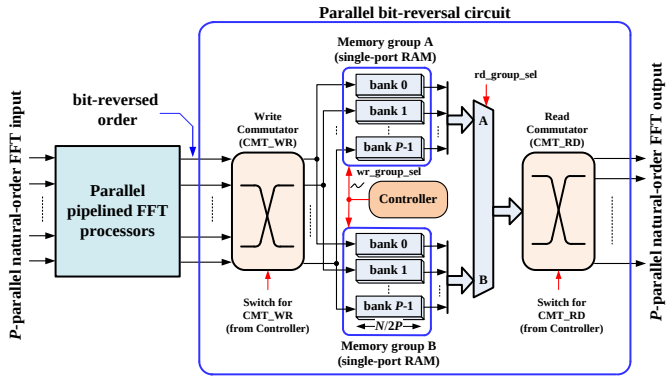

Fig. 3. Proposed parallel bit-reversal circuit.

A. Switching mechanism:

The switching mechanism is based on the idea that the P parallel inputs should be written into different banks. Likewise, the P parallel outputs must be read from different banks. In order to guarantee this, a switching mechanism is devised as follows. The switching patterns of write commutator for 4-parallel and 8-parallel paths are shown in Fig. 4 (a) and (b), respectively. Under switching pattern J, the destination bank index for output from path i, given a P-parallel architecture can be derived through modulo operation over P.

$$\text{Destination\_bank}(i, J) = \text{mod}((i+J), P). \quad (2)$$

For example, consider the structure of 4-parallel paths, when switching pattern is 3, the path 2 output will be written to memory bank 1, i.e., due to the operation of mod (2+3, 4) = 1. As shown in Fig. 2, the adjacent P $X(k)$s in a set will be stored in different memory banks by changing the switching patterns in every N/P^2 (i.e., 2^{m-2q}) cycles. The switching pattern is arranged as $\{BR(0), BR(1), \ldots, BR(P\text{-}1)\}$, which follows the bit-reverse form of q-bit binary representation. Hence, the switching pattern $J(t)$ at clock cycle t can be derived as:

$$J(t) = BR\left(\left\lfloor \frac{t}{N/P^2} \right\rfloor\right), \quad (3)$$

where $\lfloor \square \rfloor$ is the floor function. Although other switching patterns are feasible, the proposed pattern is much easier for overall design according to our extensive experiments. The commutator CMT_RD operates in a similar way to CMT_WR, except the difference that it is to switch the output from memory bank b to proper output path based on the following formula:

$$\text{Destination\_path}(b, J) = \text{mod}((b+P-J), P) \quad (4)$$

For general N and P, the detailed Write Commutator and Read Commutator architectures are shown in Fig. 5 (a) and Fig. 5(b), respectively.

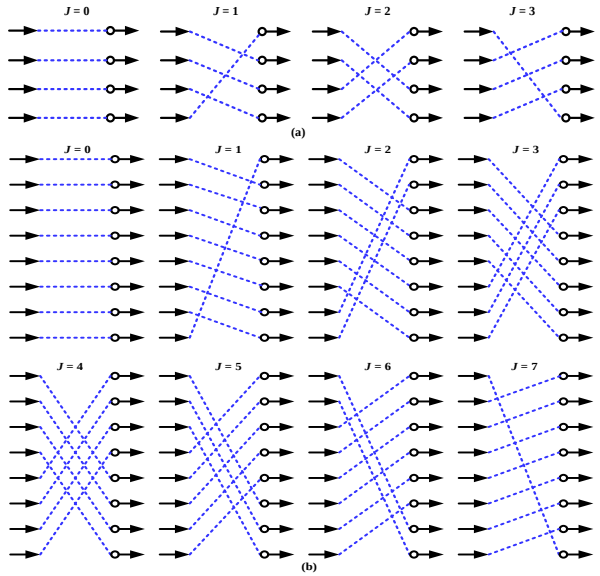

Fig. 4. Switching patterns of the proposed write commutator (a) 4-parallel case (b) 8-parallel case.

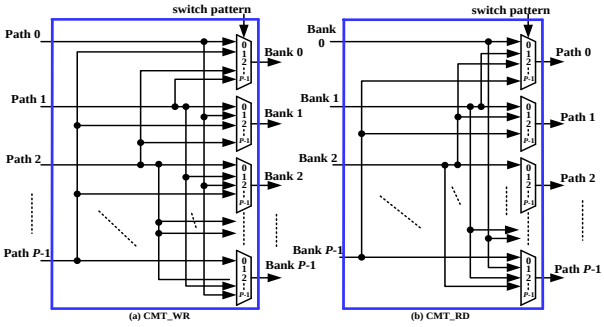

Fig. 5. Commutator architectures (a) CMT_WR (b) CMT_RD.

B. General scheduling rule for Read/Write Operations:

To access the two memory groups efficiently under continuous-flow FFT operation, the selection of memory group for write or read operations at each clock cycle should be well scheduled. The proposed scheduling mechanism can be summarized as two types for all power-of-2 FFT lengths, depending on N and P. Let $\alpha = \log_2(N/2P^2)$, and denote α as $\alpha = 2\beta$ or $\alpha = 2\beta+1$, if it is an even integer or odd integer, respectively, where β is an integer. The memory write/read scheduling of two memory groups for even-value α is shown in Fig. 6(a), while the scheduling mechanism for odd-value α is shown in Fig. 6(b). Without loss of generality, FFT output from different symbols are assumed for continuous-flow operation and each symbol period T is equal to N/P, because P samples are generated per clock cycle. First, for the first case, in the first 2^β clock cycles, the permuted data after write commutator are written into memory group A, and then followed by the data writing into memory group B in the next 2^β clock cycles. Such scheduling will be repeated periodically. During the last 2^β clock cycles of symbol 1 period, the controller will start the FFT output process by reading data from memory group A in natural order, i.e., start from $X(0) \sim X(P\text{-}1)$. The released memory space will then be available for storing the permuted FFT output of the second symbol after 2^β clock cycles. Those procedures will be applied to memory group B similarly. For the case of odd α, in the first 2^β clock cycles, the permuted data after write commutator are written into memory group A, and then followed by the data writing into memory group B in the next $2^{\beta+1}$ clock cycles. Similarly, the controller will start the read process in clock cycle $N/P - 2^{(\beta+1)}$ of symbol 1. The released memory locations will be reused by the next symbol $2^{(\beta+1)}$ clock cycles later. With the above seamless scheduling, the two groups of memory banks act as cycle-based ping-pong buffers, instead of conventional symbol-based ping-pong buffers. Hence, the memory space can be utilized very efficiently with smaller single-port memory of size N, as compared with conventional designs with larger $2N$ memories.

C. Address Generations:

The write/read address generation for the proposed parallel bit-reversal circuit is very simple and regular. Based on the previous discussion, address generation can be derived based on a cycle counter *cnt_c*. For FFT length N, *cnt_c* counts from 0 to N/P-1. Assume that the counter value is represented in (m-q)-bit binary form, as ($c_{m-q-1}, \ldots, c_1c_0$), where m, q are defined in Section III. The write address generation differs for odd and even symbols. Assuming symbol is counted from 1, then for odd symbols, the permuted data after write commutator will be written into each memory banks of group A or group B starting from address 0, and incremented by 1 for each following write operations on that group. By referring to the read/write scheduling timing diagram shown in Fig. 6, the write address of memory bank b for either group A or group B in an odd-symbol period can be represented as

$$A_{odd}^{(b)} = (c_{m-q-1}, \ldots, c_{\beta+1}, c_{\beta-1}, \ldots, c_0) \quad (5)$$

In contrast, for even symbols, the permuted data after write commutator will be written into the locations of their bit-reversed counterparts in previous symbol. Hence, the addresses for an even symbol can be derived by first computing the addresses of their counterparts in the previous symbol, followed by switching those addresses to suitable memory banks by a commutator. The switching mechanism of the commutator here is the same as the write commutator. Since the output after the write commutator from the ($p_{q-1}\ldots p_1p_0$)-th parallel path is the ($c_{m-q-1}\ldots\ c_1c_0p_{q-1}\ldots\ p_1p_0$)-th output of the current symbol, where $p_{q-1}, \ldots, p_1, p_0 \in \{0,1\}$, its bit-reversed counterpart in the previous symbol has index ($p_0p_1\ldots p_{q-1}c_0c_1\ldots c_{m-q-1}$). Based on (2), the associated data will be written to memory bank with bank index mod(J+BR($p_0p_1\ldots p_{q-1}$), P), where $J = (c_{m-2q}\ldots c_{m-q-2}c_{m-q-1})$ is its

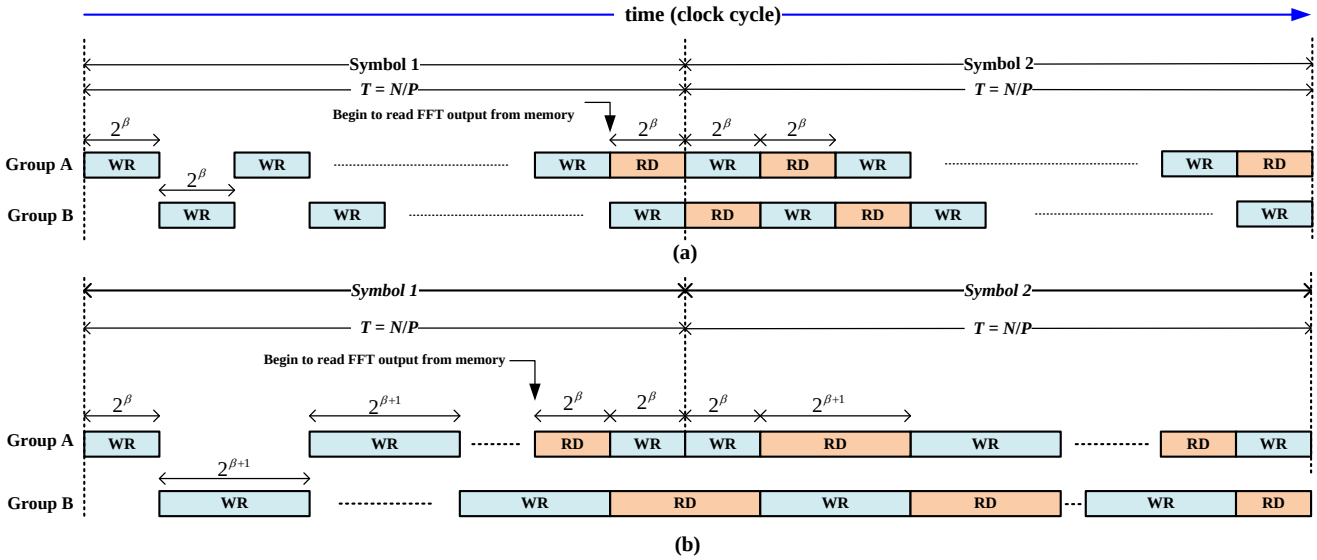

Fig. 6. General schedule mechanism for write/read operations (a) when α is even (b) when α is odd.

output path. However, J is also the switching pattern of current symbol. Hence, the write address of memory bank b in the even-symbol period can be represented as

$$A_{even}^{(b)} = (c_0 c_1, ..., c_{\alpha-\beta-1}, c_{\alpha-\beta+1}, ..., c_\alpha) + BR(\mathrm{mod}(J-b-P, P)) \times 2^\alpha \quad (6)$$

The control signal wr_group_sel selects a specific memory group to write in each clock cycle, according to the following rule:

$$\text{wr\_group\_sel} = \begin{cases} c_\beta, & \text{for even } \alpha \\ c_\beta \ xor \ c_{\beta+1}, & \text{for odd } \alpha \end{cases} \quad (7)$$

The data is written to memory group B when wr_group_sel is 1 , else it is written to memory group A. For demonstration, the write address generations are derived for 8-parallel architecture and realized, as shown in Fig. 7, which can handle FFT lengths ranging from 128 to 32768 points. In the figure, $(c_{11}c_{10}c_9 \ldots c_3c_2c_1c_0)$ is the 12-bit binary representation of *cnt_c*; and Fig. 7 (a) and Fig. 7 (b) show the write address of each memory bank for odd and even symbols, respectively. Since the read address is just the delayed copy of the write address, its derivation is omitted here. The final address for each group can be derived by multiplexing the write address and read address because single-port RAM is used. In the following, several design examples will be provided for better understanding of the proposed switching and scheduling mechanisms.

D. 8-parallel FFT Examples:

1) 128-point FFT: Without loss of generality, consider the example of 8-parallel 128-point FFT with continuous-flow operation. The output sequence $X(k)$ from FFT processor for two contiguous symbols are shown in Fig. 8(a). Commutator CMT_WR will switch the sequence every two clock (i.e., $128/(8^2) = 2$) cycles. The switching pattern is {0,4,2,6,1,5,3,7}, which repeats again for the second symbol. The permuted sequence after CMT_WR is shown in Fig. 8(b). The scheduling of write/read operations is shown in Fig. 8(c). At clock cycle 15, the last output set {$X(79)$,$X(47)$,…,$X(15)$} of symbol 1 is written into memory group B, meanwhile the first natural-order output set {$X(0)$, $X(1)$, $X(2)$, $X(3)$, $X(4)$, $X(5)$, $X(6)$, $X(7)$} of symbol 1 is read from memory group A; and those memory space can be released for the incoming symbol 2 later. Therefore, at clock cycle 16, the first output set {$X(0)$,$X(64)$, $X(32)$,$X(96)$,$X(16)$,$X(80)$,$X(48)$,$X(112)$} of symbol 2 is written into those corresponding released locations in memory group A. Similar procedures are applied to all the other output $X(k)$ sequences of symbol 2. Fig. 8(d) shows the distribution of all $X(k)$s of symbol 1 in the two memory groups right after clock cycle 15, while Fig. 8(e) shows the distribution of all $X(k)$s of symbol 2 in the memory groups after clock cycle 31. It is interesting to note that each $X(k)$ of symbol 2 is stored in the location of its bit-reversed counterpart of symbol 1.

2) 8-parallel 256-point FFT: Similarly, the bit-reversed output $X(k)$ sequences of 256-point FFT are shown in Fig. 9(a). CMT_WR switches those sequences every four clock cycles according to the same switching pattern. The permuted sequences after CMT_WR are shown in Fig. 9(b). However, the scheduling of write/read operation is different from that of 8-parallel 128-point FFT. In clock cycle 0, the first output set {$X(0)$,$X(128)$, …, $X(224)$} of symbol 1 is written into memory group A, followed by the memory write of the 2nd and the 3rd sets of symbol 1 into memory group B in cycle 1 and cycle 2, respectively. Then the 4th and the 5th set will be written to group A again, and these procedures are applied to the following output sequences again. In clock 30, the first natural-order output set {$X(0)$,$X(1)$, …, $X(7)$} of symbol 1 is scheduled to be read out from memory group A, followed by the readout of the 2nd set (i.e., {$X(8)$,$X(9)$, …, $X(15)$}) and the 3rd set (i.e., {$X(16)$, $X(17)$, …, $X(23)$}) from memory group B in clock cycle 31 and clock cycle 32, respectively. Then the read operations are switched back to memory group A again. In clock cycle 32, the first $X(k)$ set of symbol 2 arrives which is stored in the released space set {$X(0)$,$X(1)$, …,$X(7)$} of symbol 1 previously. This procedure is again applied to the following incoming sequences, i.e., those sequences will be written into the released memory locations two clock cycles ago. The distribution of all $X(k)$s of symbol 1 in the two memory groups after clock cycle 31 is shown in Fig. 9(d), while the distribution of all $X(k)$s of symbol 2 after clock cycle 63 is shown in Fig. 9(e). Obviously, there are other possible scheduling approaches, for example, the scheduling shown in Fig. 10, where the first two output sets are written to memory group A, followed by two output sets written into group B. Under this arrangement, read operations should be scheduled for group A in clock cycles 30 and 31. However, $X(8)$ was stored in group B at clock cycle 2. It means that one should read $X(8)$ from group B in clock 31, which violates the pre-scheduled write operation of group B in clock cycle 31, because single-port memory is assumed. Therefore, such scheduling is not allowed.

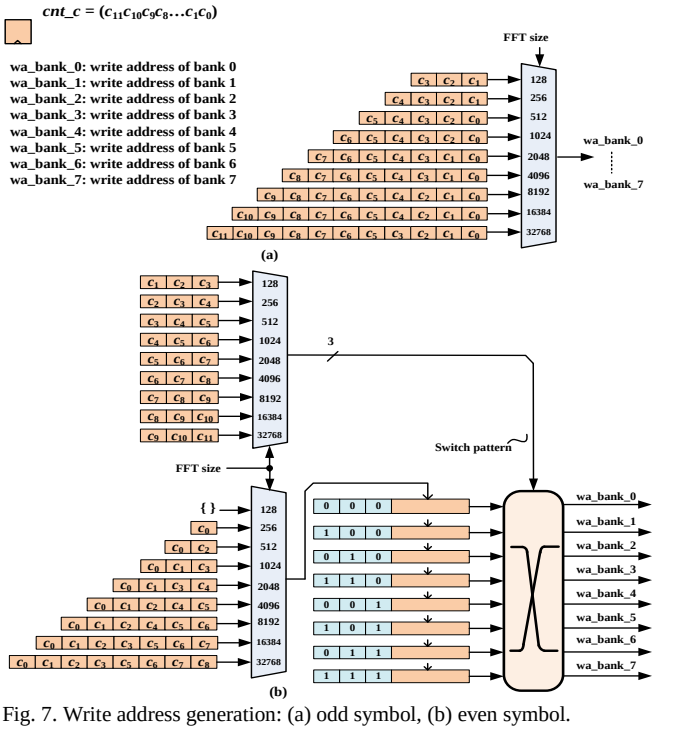

Fig. 7. Write address generation: (a) odd symbol, (b) even symbol.

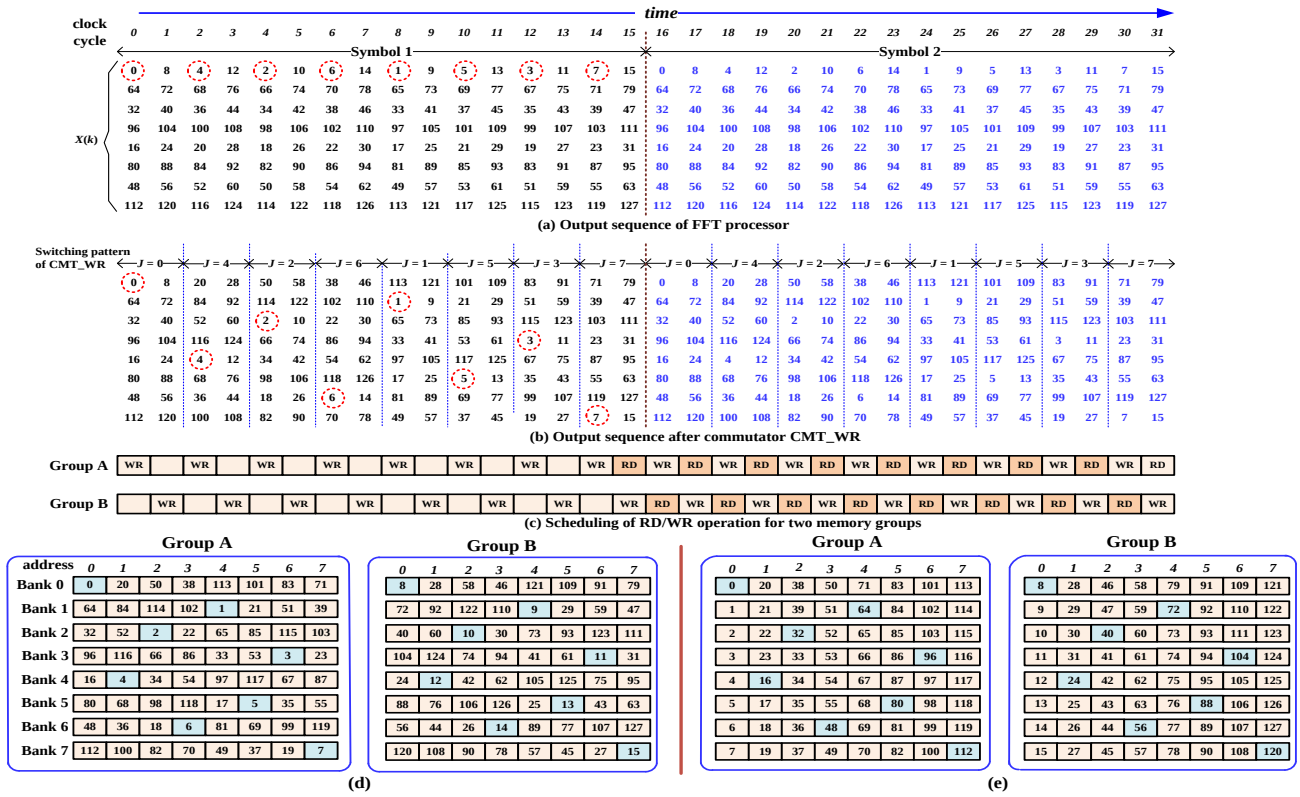

Fig. 8. Example of 8-parallel 128-point FFT. (a) output sequences from FFT processor (b) permuted sequences after commutator CMT_WR (c) the scheduling of write/read operations (d) the distributions of $X(k)$ in memory banks after the 15th cycle (e) the distributions of $X(k)$ in memory banks after the 31th cycle.

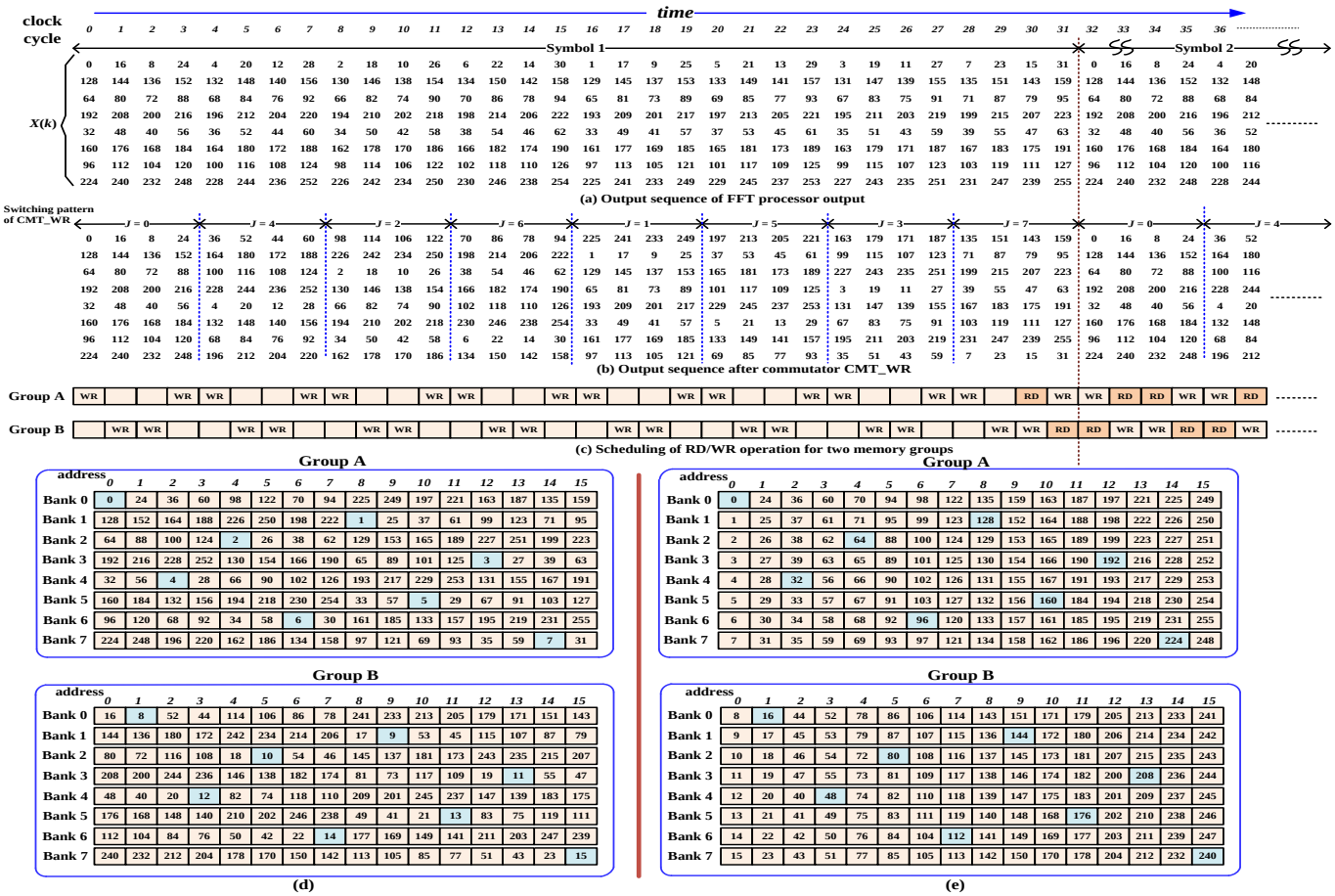

Fig. 9. Example of 8-parallel 256-point FFT (a) original sequences from FFT processor (b) permuted sequences after commutator CMT_WR (c) the scheduling of write/read operations (d) the distribution of $X(k)$ in eight memory banks after the 31th cycle (e) the scheduling of $X(k)$ in eight memory banks after the 63th cycle.

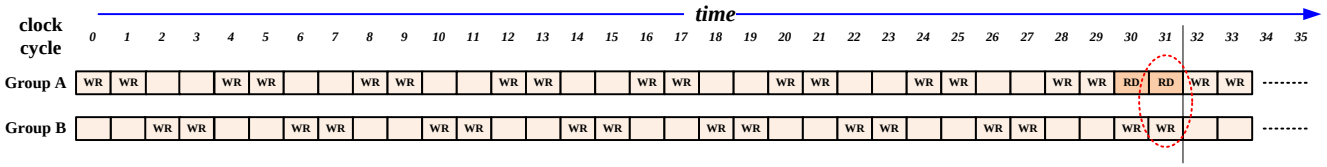

Fig. 10. An example of failed scheduling approach for 8-parallel 256-point FFT.

V. Implementations and Comparisons with Existing Works

The proposed parallel bit-reversal architecture can support general power-of-2 FFT lengths. To verify its correctness for continuous-flow operation, the simulation patterns of bit-reversed FFT output are generated by Matlab first, and then loaded as reversal circuit's input when simulation begins. Simulation results verify that the proposed scheme is correct for FFT lengths ranging from 128 to 32768 points for 2-parallel, 4-parallel, and 8-parallel architectures. Based on scheduling shown in Fig. 6, the latency for natural-order output is $(N/P)-2^{\beta}+1$, and $(N/P)-2^{(\beta+1)}+1$ clock cycles for even α and odd α, respectively. The throughput is P samples per clock cycle. The 8-parallel realization of the proposed bit-reversal circuit, which supports 128 to 32768-point FFT, is accomplished with TSMC-90nm process. Its memory unit is realized with 16 single-port SRAM macros with data width of 32 bits. The pre-layout gate-level synthesis results show that the whole cell area is 1905852 μm^2, including logic part area of 11641 μm^2 and memory area of 1894211 μm^2. It is observed that the logic part area is relatively small compared to the memory area. Its power consumption is 2.017 mW when it is operated at 320 MHz clock frequency.

The comparisons of existing bit-reversal circuits and their associated features are shown in Table I. The first column lists the circuits designed to calculate the bit reversal. The second column lists the type of FFT architectures for which they calculate the bit reversal. The third column shows the parallelism degree of each design. For SDF and SDC FFT architectures, the reordering circuits only process serial data, whereas for MDC and MDF FFT architectures the reordering circuits handle several parallel data simultaneously. The throughput data shown in the last column corresponds to their respective architectures. Note that serial bit-reversal circuit processes one sample per clock cycle, whereas parallel ones process P samples in parallel per clock cycle. The fourth column of the table shows the FFT output pattern types presented to the reordering circuits. Most of the compared designs perform bit-reversal operation. However, some of them do not expect data in bit-reversed order from the FFT module, but in another specific order or pattern. Finally, the fifth column indicates the sizes and types of the memories needed to facilitate the reordering of data samples, and the sixth column shows whether those designs support continuous-flow operations and variable FFT length or not.

From Table I, it can be observed that the reordering circuits [3-5], [11], [17-19] are only for serial data, and not applicable to parallel MDC and MDF FFTs. The works in [3-5] propose modified SDC FFT architectures. Their bit-reversal circuits are merged with the last-stage butterfly unit of FFT processors. The bit-reversal function is achieved with extra data scheduling. The bit-reversal circuit in [17] requires memory size of $2N$. In [11] the bit reversal for real-valued FFTs is calculated, which is a specific order different to the bit-reversal in conventional FFTs. In [18], the minimum buffer and latency required to reorder the input data are derived by mathematical analysis. In [19], the bit-reversal circuit is composed of simple buffers and multiplexers. This work provides the optimum bit reversal circuits designs for serial data for radix-2, radix-2^k, radix-4 and radix-8. In case of parallel data, the works in [9-10], [12] target pipelined MDC FFT processors. Among them, the work in [9] only targets the case of 8-parallel data. This design is costly in terms of memory, as it requires a total memory size of $P{\cdot}N$. The work in [10] presents a more efficient approach, as it requires slightly more than N memory words by using several sporadic small-size FIFOs. However, this design is only suited for a specific FFT output order pattern, instead of bit-reversed FFT output pattern. Compared to those works, the proposed approach targets FFT outputs of parallel data in bit-reversed order (not in other specific unconventional orders) and uses only a total memory size of N words. Another alternative in [12] calculates the bit reversal for parallel data using approximately the same memory as the proposed approach. However, the detail of the bit-reversal circuit that carries out the reordering is not described. As such, the proposed design is the first circuit that calculates the bit reversal algorithm for parallel data using only a total memory size of N words, and in particular, only single-port RAM is used, instead of two-port RAM adopted by all the compared designs. Take 8-parallel case for example, the area comparisons between a single-port 32-bit RAM and a two-port 32-bit RAM under different FFT sizes for both 90-nm and 55-nm processes are listed in Table II. Since no two-port synchronous SRAM are provided by our memory compiler tool, single-port Register File and two-port Register File are chosen for comparisons. For each FFT size, in addition to the area data, the table also shows the area ratio (in percentage) of the single-port RAM over the two-port RAM, where the two-port RAM is set as 100 %. As shown, a larger FFT size has better area reduction ratio than a smaller FFT size, which can be up 50 %, while at least around 30 % area reduction can be obtained for 2048-point FFT.

VI. Conclusion

In this work, a new parallel bit-reversal circuit is proposed for parallel MDF and MDC pipelined FFT processors. The proposed architecture is cost-effective because only single-port RAM of total size N is required for N-point continuous-flow FFT. Besides, the addressing scheme is simple and regular for all power-of-2 FFT lengths, and it supports variable length processing. For future work, it is a very challenging task to further improve the proposed architectures so that the required

memory space can be less than *N*. In addition, generalization of the proposed design techniques to MIMO FFTs with very high throughput is also a good research direction.

TABLE I. COMPARISONS OF SEVERAL BIT REVERSAL CIRCUITS

	Supported FFT Architecture	Supported Data Parallelism (*P*)	Input Data Pattern to the Reordering Circuit	Reordering Memory		Continuous-flow / variable length support	Throughput (samples/ cycle)	Latency (cycles)
				Size (words) / port number	No. banks			
[18]	SDF	Only serial data	Bit-reversed	N / two-port	1	Yes/No	1	L_18[a]
[19]	SDF	Only serial data	Bit-reversed	$(\sqrt{N}-1)^2$ / two-port	1	Yes/No	1	$(\sqrt{N}-1)^2$
[17]	SDF	Only serial data	Bit-reversed	$2N$ / two-port	2	Yes/No	1	Not shown
[3-5]	SDC (modified)	Only serial data	Specific pattern	N / two-port	1	Yes/No	1	Not shown
[2]	SDF (modified)	Only serial data	Specific pattern	$N/2$ / shift registers	4	Yes/No	1	78[b]
[11]	Real-valued FFT	Only serial data	Bit-reversed real-valued data	$(5N/8)$-3 / two-port	1	Yes/No	1	$(5N/8)$-3
[12]	MDC (modified)	Parallel (2/4/8/16)	Bit-reversed	N-(N/P) / two-port	Not shown	Yes/No	P	Not shown
[10]	MDC	Parallel (8)	Specific pattern (set-reversed)	$(9/8)N$+192 / two-port	8+3+3+6×4	Yes/No	P	Not shown
[9]	MDC	Parallel (8)	Bit-reversed	$P \cdot N$ / two-port	8	Yes/No	P	Not shown
This work	MDF or MDC	Parallel (2/4/8)	Bit-reversed	N / single-port	$2P$	Yes/Yes	P	L_26[c]

a: L_18 = $(2^{n/2}-1)^2$ for even n and $(2^{(n+1)/2}-1)(2^{(n-1)/2}-1)$ for odd n, where $n = \log_2(N)$.

b: 78 includes FFT operation time for 64-point FFT.

c: L_26 = $(N/P) - 2^{\beta} + 1$ for even α and $(N/P) - 2^{(\beta+1)} + 1$ for odd α.

Table II. Memory area comparisons for different FFT sizes using single-port RAM and two-port RAM

Memory / FFT size	90nm Process (area unit: mm^2)		55nm Process (area unit: mm^2)	
	two-port Register File	single-port Register File	two-port Register File	single-port Register File
32768	No macro with depth 4096 provided	0.089×16	No macro with depth 4096 provided	0.048×16
16384	0.188×8 (100 %)	0.051×16 (54.2 %)	0.096×8 (100 %)	0.023×16 (47.9 %)
8192	0.11×8 (100 %)	0.032×16 (58.2 %)	0.052×8 (100 %)	0.013×16 (50 %)
4096	0.07×8 (100 %)	0.023×16 (65.7 %)	0.029×8 (100 %)	0.008×16 (55.2 %)
2048	0.051×8 (100 %)	0.018×16 (70.5 %)	0.017×8 (100 %)	0.005×16 (58.8 %)

REFERENCES

[1] S. He and M. Torkelson, "Designing pipeline FFT processor for OFDM (de)modulation," in *Proc. URSI Int. Symp. Signals, Syst., Electron*., pp. 257-262, 1998.

[2] S. Lee and S.C. Park, "A modified SDF architecture for mixed DIF/DIT FFT," *IEEE Int. Symp. Circuits and Systems*, pp. 2590-2593, 2007.

[3] Y. N. Chang "An efficient VLSI architecture for normal I/O order pipeline FFT design," *IEEE Trans. Circuit and Systems-II*, vol. 55, issue. 12, pp. 1234-1238, 2008.

[4] Y. N. Chang "Design of an 8192-point sequential I/O FFT Chip," *Proceedings of the world congress on engineering and computer sciences(WCECS)*, vol. II, 2012.

[5] Xue Liu, Feng Yu, and Ze-ke Wang, "A pipelined architecture for normal I/O order FFT," *Journal of Zhejiang University-Science C*, pp. 76-82, June 2011.

[6] Y.W. Lin, H.Y. Liu, 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.

[7] M. Shin and Hanho Lee, "A high-speed four-parallel radix-2^4 FFT/IFFT processor for UWB applications," *IEEE Int. Symp. Circuits and Systems*, pp. 960-963, 2008.

[8] S.N. Tang, J.W. Tsai, and T.Y. Chang, "A 2.4-GS/s FFT processor for OFDM-Based WPAN applications," *IEEE Trans. Circuits Syst. II*, vol. 6, no. 57, pp. 451-455, June. 2010.

[9] S. Yoshizawa, A. Orikasa, and Y. Miyanaga, "An area and power efficient pipeline FFT processor for 8x8 MIMO-OFDM systems," *IEEE Int. Symp. Circuits and Systems*, pp. 2705-2708, 2011.

[10] Kai-Jiun Yang, Shang-Ho Tsai, and Gene C.H. Chuang, "MDC FFT/IFFT processor with variable length for MIMO-OFDM Systems," *IEEE Trans. VLSI*, vol. 21, no. 4, pp. 720-731, 2013.

[11] M. Ayinala, M. Brown, and KK. Parhi, "Pipelined parallel FFT architectures via folding transforms," *IEEE Trans. VLSI*, vol. 20, no. 6, pp. 1068-1081, 2012.

[12] M. Garrido, J. Grajal, M.A. Sanchez, and O. Gustafsson, "Pipelined radix-2^k feedforward FFT architectures," *IEEE Trans. VLSI*, vol. 21, no. 1, pp. 23-32, 2013.

[13] E. H. Wold and A. M. Despain, "Pipeline and parallel-pipeline FFT processors for VLSI implementations," *IEEE Trans. Comput*., vol. C-33, no. 5, pp. 414-426, May 1984.

[14] Sorokin, H and Takala, J, "Conflict-free parallel access scheme for mixed-radix FFT supporting I/O permutations," *IEEE Int. Conference on Acoustics, Speech, and Signal Processing*, pp. 1709-1712, 2011.

[15] S.J. Huang and S.G. Chen, "A high-throughput radix-16 FFT processor with parallel and normal input/output ordering for IEEE 802.15.3c systems," *IEEE Trans. Circuit and Systems-I*, vol. 59, issue. 8, pp. 1752-1765, 2012.

[16] H. S. Hu, H.Y. Chen, and Shyh-Jye Jou, "Novel FFT processor with parallel-in-para llel-out in normal order," *Int. Symp. VLSI Design, Automation and Test*, pp. 150-153, 2009.

[17] F. Kristensen, P. Nilsson, and A. Olsson, "Flexible baseband transmitter for OFDM," in *Proc. IASTED Conf. Circuits Signals Syst.*, May 2003, pp. 356–361.

[18] T. S. Chakraborty and S. Chakrabarti, "On output reorder buffer design of bit-reversed pipelined continuous data FFT architecture," *IEEE Asia Pacific Conference on Circuits and Systems* (*APCCAS*), pp. 1132-1135, 2008.

[19] M. Garrido, J. Grajal, and O. Gustafsson "Optimum Circuits for Bit Reversal," *IEEE Trans. Circuit and Systems-II*, vol. 58, issue. 10, pp. 657-661, 2011.
[20] D. Sundararajan, M. Omair Ahmad, and M. N. S. Swamy, "A fast FFT bit-reversal algorithm," *IEEE Trans. Circuit and Systems-II: Analog and digital signal processing*, vol. 41, no. 10, pp. 701-703, 1994.
[21] Juan M. Rius, and R. De Porrata-Doria, "New FFT bit-reversal algorithm," *IEEE Trans. Signal Processing*, vol. 43, no. 4, pp. 991-994, 1995.
[22] J. Prado, "A New fast bit-reversal permutation algorithm based on symmetry," vol. 11, no. 12, pp. 933-936, 2004.
[23] M. A. Jaber, and D. Massicotte, "A novel approach for FFT data reordering," *IEEE Int. Symp. Circuits and Systems*, pp. 1615-1618, 2010.
[24] T.C. Choinski, and T. T. Tylaska, "Generation of digit reversed address sequences for fast fourier transforms," *IEEE Trans. Computers*, vol. 40, no. 6, pp. 780-784, 1991.
[25] Seung Ho Ok, and Byung In Moon, "A digit reversal circuit for the variable-length radix-4 FFT," *Future generation communication and networking*, vol. 2, pp. 496-500, 2007.

Sau-Gee Chen received his B.S. degree from National Tsing Hua University, Taiwan, in 1978, M.S. degree and Ph.D. degree in electrical engineering, from the State University of New York at Buffalo, NY, in 1984 and 1988, respectively. Currently, he is the chairman and a professor at the Department of Electronics Engineering, National Chiao Tung University (NCTU), Taiwan. He serves as the IEEE VTS Asia-Pacific Chapters chair, starting from 2014. He was the Chair, IEEE Vehicular Technology Society, Taipei Chapter, during 2012-2013. He was Director of Honors Program, College of Electrical & Computer Engineering/College of Computer Science from 2011 to 2012 at NCTU. He also was the Associate Dean, Office of International Affairs, during March-July, 2011, as well as the director of Institute of Electronics from 2003 to 2006, all at the same organization. During 2004-2006, he served as an associate editor of IEEE Transactions on Circuits and Systems I. His research interests include digital communication, multi-media computing, digital signal processing, and VLSI signal processing. He has published more than 100 conference and journal papers, and holds 14 US and Taiwan patents.

Shen-Jui Huang received his M.S. degree from the Department of Electrical Engineering, National Taiwan University in 1997, and Ph.D. degrees in electronics from National Chiao Tung University in 2012. Over the past years, he worked in Hsinchu Science Park, Taiwan, as a digital IC design engineer for Bluetooth, WiFi, and Wimax chips. Currently he is Principal Engineer in Novatek Corp., Hsinchu, Taiwan. His research interests include baseband signal processing, efficient implementation of communication IC, and reconfigurable hardware architectures.

Mario Garrido received the M.S. degree in electrical engineering and the Ph.D. degree from the Technical University of Madrid (UPM), Madrid, Spain, in 2004 and 2009, respectively. In 2010 he moved to Sweden to work as a postdoctoral researcher at the Department of Electrical Engineering at Linköping University. Since 2012 he is Associate Professor at the same department. His research focuses on optimized hardware design for signal processing applications. This includes the design of hardware architectures for the calculation of transforms, such as the fast Fourier transform (FFT), circuits for data management, the CORDIC algorithm, and circuits to calculate statistical and mathematical operations. His research covers high-performance circuits for real-time computation, as well as designs for low area and low power consumption.

Shyh-Jye Jou received his B. S. degree in electrical engineering from National Chen Kung University in 1982, and M. S. and Ph.D. degrees in electronics from National Chiao Tung University in 1984 and 1988, respectively. He joined Electrical Engineering Department of National Central University, Chung-Li, Taiwan, from 1990 to 2004 and became a Professor in 1997. Since 2004, he has been Professor of Electronics Engineering Department of National Chiao Tung University and served as the Chairman from 2006 to 2009. From August 2011 he becomes the Vice President for International Affair. He was a visiting research Professor in the Coordinated Science Laboratory at University of Illinois, Urbana-Champaign during 1993-1994 and 2010 academic years. In the summer of 2001, he was a visiting research consultant in the Communication Circuits and Systems Research Laboratory of Agere Systems, USA. He received Outstanding Engineering Professor Award, Chinese Institute of Engineers and Chinese Institute of Electrical Engineering at 2011 and 2013, respectively.
He served as 2006 Chapter Chair of IEEE Circuits and Systems Society Taipei Chapter, 2009-2010 Distinguished Lecturer of CAS society, Guest Editor, IEEE Journal of Solid State Circuits, Nov. 2008, and was Track Chair, 2011-2013Nanoelectronics and Gigascale Systems. He also serves as Conference Chairs, TPC Chairs in many IEEE conferences such as Conference Chair of IEEE VLSI-DAT and International Workshop on Memory Technology, Design, and Testing. He also served as Technical Program Chair or Co-Chair in IEEE VLSI-DAT, International IEEE Asian Solid-State Circuit Conference, IEEE Biomedical Circuits and Systems. He has published more than 100 IEEE journal and conference papers. His research interests include design and analysis of high speed, low power mixed-signal integrated circuits, communication and Bio-Electronics integrated circuits and systems.