MDC FFT/IFFT Processor With Variable Length for MIMO-OFDM Systems

Kai-Jiun Yang, Shang-Ho Tsai, *Senior Member, IEEE*, and Gene C. H. Chuang, *Member, IEEE*

***Abstract*— This paper presents an multipath delay commutator (MDC)-based architecture and memory scheduling to implement fast Fourier transform (FFT) processors for multiple input multiple output-orthogonal frequency division multiplexing (MIMO-OFDM) systems with variable length. Based on the MDC architecture, we propose to use radix-N_s butterflies at each stage, where N_s is the number of data streams, so that there is only one butterfly needed in each stage. Consequently, a 100% utilization rate in computational elements is achieved. Moreover, thanks to the simple control mechanism of the MDC, we propose simple memory scheduling methods for input data and output bit/set-reversing, which again results in a full utilization rate in memory usage. Since the memory requirements usually dominate the die area of FFT/inverse fast Fourier transform (IFFT) processors, the proposed scheme can effectively reduce the memory size and thus the die area as well. Furthermore, to apply the proposed scheme in practical applications, we let $N_s = 4$ and implement a 4-stream FFT/IFFT processor with variable length including 2048, 1024, 512, and 128 for MIMO-OFDM systems. This processor can be used in IEEE 802.16 WiMAX and 3GPP long term evolution applications. The processor was implemented with an UMC 90-nm CMOS technology with a core area of 3.1 mm^2. The power consumption at 40 MHz was 63.72/62.92/57.51/51.69 mW for 2048/1024/512/128-FFT, respectively in the post-layout simulation. Finally, we analyze the complexity and performance of the implemented processor and compare it with other processors. The results show advantages of the proposed scheme in terms of area and power consumption.**

***Index Terms*— 3GPP, 802.16, fast Fourier transform (FFT), long term evolution (LTE), memory scheduling, multiple-input multiple-output (MIMO), orthogonal frequency division multiplexing (OFDM), output sorting, pipeline multipath delay commutator (MDC), WiMAX.**

I. Introduction

FAST Fourier transform (FFT) is a crucial block in orthogonal frequency division multiplexing (OFDM) systems. OFDM has been adopted in a wide range of applications from wired-communication modems, such as digital subscriber lines (xDSL) [1], [2], to wireless-communication modems, such as IEEE802.11 [3] WiFi, IEEE802.16 [4], [5] WiMAX or 3GPP long term evolution (LTE), to process baseband data. Inverse fast Fourier transform (IFFT) converts the modulated information from frequency domain to time domain for transmission of radio signals, while FFT gathers samples from the time domain, restoring them to the frequency domain. With multiple input multiple output (MIMO) devices, data throughput can be increased dramatically. Hence MIMO-OFDM systems provide promising data rate and reliability in wireless communications [6]. To handle "multiple" data streams, intuitively the functional blocks need to be duplicated for processing the concurrent inputs. Without a proper design, the complexity of FFT/IFFT processors in MIMO systems grows linearly with the number of data streams.

In-place-memory-updating and pipelines are the architectures most widely adopted for the implementation of FFT/IFFT. From the memory access perspective, in-place memory updating schemes performs the computation in three phases: writing in the inputs, updating intermediate values, and reading out the results. In updating phase, the processor reuses the radix-r processor, such that a single radix-r butterfly is sufficient to complete N-point FFT/IFFT computation. Since each phase is non-overlapped, the outputs can be sequential or as requested. However, it is the non-overlapping characteristic that makes the butterfly idle in memory write and read phases, and the overall process is lengthy. Continuous-flow mixed-radix (CFMR) FFT [8], [9] utilizes two N-sample memories to generate a continuous output stream. One of the memories is used to calculate current FFT/IFFT symbols, while the other stores the previously computed results and controls the output sequence. Thus, when CFMR is used in MIMO systems, the required memory is increased in a trend proportional to $2 \times N_s$, where N_s is the number of data streams. Such memory requirement may be forbidden if N_s is large, because the area of memory does not shrink as much as that of logic gates when fabrication technology advances, due to the use of sense amplify circuitry.

As for pipeline schemes, single-path delay feedback (SDF) and multipath delay commutator (MDC) are the two most popular architectures [10]. Cortes *et al.* proposed a procedure to decompose a discrete Fourier transform matrix so that the FFT processor can be implemented with pipeline systematically [11]. SDF schemes provide feedback paths to manage partially computed results in each pipe and to generate seamless output without delay. The first output sample can be generated immediately after the last input sample has been fed into the FFT/IFFT processor. Furthermore, with the scheduling

Manuscript received September 23, 2011; revised January 26, 2012; accepted March 21, 2012. Date of publication May 4, 2012; date of current version March 18, 2013. This work was supported in part by the Aim for the Top University Plan of the National Chiao Tung University and Ministry of Education, Taiwan, and the National Science Council, Taiwan, under Grant NSC 99-2221-E-009-100-MY3.

K.-J. Yang and S.-H. Tsai are with the Department of Electrical Engineering, National Chiao Tung University, Hsinchu 300, Taiwan (e-mail: kaijiuny.ece98g@g2.nctu.edu.tw; shanghot@mail.nctu.edu.tw).

G. C. H. Chuang is with the Industrial Technology Research Institute, Hsinchu 31015, Taiwan (e-mail: genechuang@itri.org.tw).

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

Digital Object Identifier 10.1109/TVLSI.2012.2194315

of input data, SDF schemes are capable of processing multiple input streams using a single FFT/IFFT processor [12], [14]. On the other hand, MDC schemes parse feedback paths into feed forward streams using switch-boxes with more memory [15]. Meanwhile, the radix-r butterflies idle until the rth input is in position. Although the control of data flow in MDC is more straightforward, the utilization rate of the MDC FFT/IFFT computing core is $1/r$, which is far less than the 100% utilization rate in SDF FFT/IFFT. Sansaloni *et al.* suggested that MDC could save more area than SDF in FFT with multiple streams [16], and Fu implemented a four-stream MDC FFT/IFFT processor in which the area was 75% that of conventional designs [20]. To obtain parallelism, radix-2 butterflies were duplicated at the first stage. Together with storage elements, generally the first module occupied the largest area.

To the best of our knowledge, for the FFT/IFFT processors used in MIMO-OFDM systems, most of the researches intuitively duplicated the butterflies and memory according to the number of data streams, and then sought ways to maximize parallelism while reducing the hardware complexity. Also, few works have considered output memory needed for bit-reversed reordering for MIMO FFT/IFFT processors. These motivate us to explore an FFT/IFFT architecture for MIMO systems, which can easily achieve a 100% utilization rate while the control mechanism is still simple. Meanwhile, we would like to reduce the memory requirement for managing bit/set-reversed output order in the new architecture.

In this paper, we consider MIMO-OFDM systems with N_s data streams, and propose to use single radix-N_s butterfly at each folding stage to implement an MDC MIMO FFT/IFFT processor. In conventional radix-r MDC FFT/IFFT processor with single data stream, the utilization rate is $1/r$. Hence, $(r-1)/r$ computing resource and memory are wasted. However, for a MIMO-OFDM system with N_s data streams, if we let $r = N_s$, the vacancy can be filled and thus the processor can achieve a 100% utilization. It is worthwhile to emphasize that by doing it we only need one butterfly at each pipeline stage. Since we use one butterfly to process N_s data streams at each pipeline stage, the input data need to be well scheduled before passing to the processor. Thanks to the simple control mechanism of MDC, we propose a simple mechanism for input scheduling, where the mechanism is scalable for N_s being power of 2. Moreover, due to the use of one butterfly at each stage, we propose a simple output scheduling for bit/set-reversing, which can greatly reduce the required output memory. If the required output memory size is N_w for single data stream, the size remains nearly N_w for multiple streams instead of $N_w \times N_s$ in conventional schemes, where multiple butterflies are needed in each pipeline stage. Furthermore, to apply the proposed schemes in practical applications, we let $N_s = 4$ and implement a 4-stream FFT/IFFT processor with variable length including 2048, 1024, 512, and 128. This processor was implemented using an UMC 90 nm process and can be used in LTE or Wi-MAX applications.

The organization of this paper is as follows. Section II lists the FFT/IFFT algorithm for the proposed architecture. Section III introduces the memory scheduling rules and the hardware requirements to provide the proposed features at low cost. Section IV includes the hardware implementation, a synthesis report, and analysis of performance. Core area and power consumption are used to compare the proposed design with existing designs. Conclusions are provided in the last section.

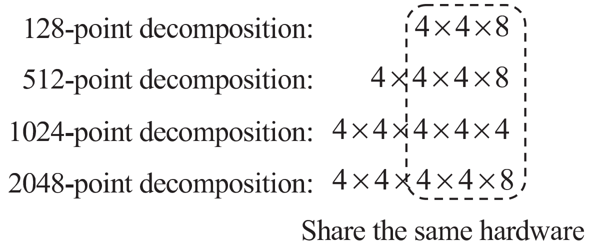

128-point decomposition: 4×4×8
512-point decomposition: 4×4×4×8
1024-point decomposition: 4×4×4×4×4
2048-point decomposition: 4×4×4×4×8
Share the same hardware

Fig. 1. Decomposition of four different FFT/IFFT lengths.

II. Algorithm

The N-point FFT and IFFT are calculated as follows:

$$X[k] = FFT\{x[n]\} = \sum_{n=0}^{N-1} x[n] W_N^{nk} \tag{1}$$

and

$$x[n] = IFFT\{X[k]\} = \frac{1}{N} \sum_{k=0}^{N-1} X[k] W_N^{-nk} \tag{2}$$

where

$$W_N^{nk} = \cos\left(\frac{2\pi nk}{N}\right) - j \sin\left(\frac{2\pi nk}{N}\right). \tag{3}$$

The IFFT can be realized by slightly modifying the FFT. That is, the IFFT of $X[k]$ can be obtained by [7]

$$x[n] = \frac{1}{N}(FFT\{X^*[k]\})^*.$$

Generally N is power of 2 and the implementation of $1/N$ only involves right shift operation. Therefore, the IFFT can share the same hardware with FFT.

In this paper, we take LTE and Wi-MAX systems as examples to implement the FFT/IFFT processor. For these two systems, there are four FFT/IFFT lengths, that is, $N = 2048$, 1024, 512, and 128. We fold the four FFT/IFFT lengths using radix-4 butterflies as many times as possible, as shown in Fig. 1. Note that the last three stages of the four FFT/IFFT lengths can share the same hardware.

Based on the decomposition in Fig. 1, (1) can be rewritten as

$$X[K] = \sum_{n_5=0}^{7} \left\{ \sum_{n_4=0}^{3} \left\{ \sum_{n_3=0}^{3} \left\{ \sum_{n_2=0}^{3} \left\{ \sum_{n_1=0}^{3} x[N] W_4^{n_1k_1} W_{2048}^{N_1k_1} \right\} \times W_4^{n_2k_2} W_{512}^{N_2k_2} \right\} W_4^{n_3k_3} W_{128}^{N_3k_3} \right\} W_4^{n_4k_4} W_{32}^{n_5k_4} \right\} W_8^{n_5k_5} \tag{4}$$

where $K = k_1 + 4k_2 + 16k_3 + 64k_4 + 256k_5, k_1 = 0 \sim 3$, $k_2 = 0 \sim 3, k_3 = 0 \sim 3, k_4 = 0 \sim 3, k_5 = 0 \sim 7$, and $N = 512n_1 + 128n_2 + 32n_3 + 8n_4 + n_5, N_1 = 128n_2 + 32n_3 + 8n_4 + n_5, N_2 = 32n_3 + 8n_4 + n_5$, and $N_3 = 8n_4 + n_5$. Each brace includes computations of a radix-4 butterfly and a twiddle factor multiplication. For non-power-of-4 FFT/IFFT,

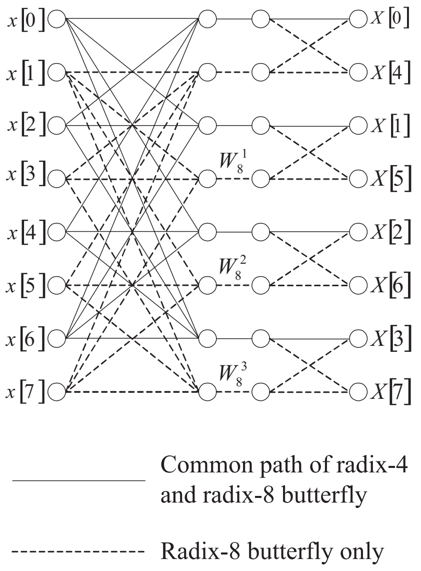


Fig. 2. SFG of the proposed radix-4/radix-8 butterfly.

a radix-8 butterfly is placed at the last stage. Hence, the last stage is configurable for both radix-4 and radix-8 computation. The proposed radix-4/radix-8 butterfly for the last stage is shown in Fig. 2, where a radix-8 butterfly has the data path indicated by both solid and dashed lines, whereas a radix-4 butterfly has the data path indicated by solid lines only. The regularity of the decomposition makes the processor scalable. This means parameterized register-transfer-level source code is highly reusable to extend the number of stages for a large N.

III. MDC Architecture for MIMO FFT/IFFT

Storage elements dominate most of the area in conventional MDC architecture. That is, the input buffering stage for radix-4 based FFT/IFFT needs $N/4 + N/2 + 3N/4$ words of memory, and each computing stage needs $3N/4^s$ words of memory, where s is the stage index. For a 2048-point MDC FFT/IFFT processor, 5112 words of memory are required. If MDC is applied in MIMO-OFDM systems, the memory size grows linearly with the number of data streams. As for the utilization rate of butterflies and multipliers, since $3/4$ of the computing time is used to gather the input data, the utilization rate is only 25% in single stream radix-4 MDC FFT/IFFT. Although MDC architecture offers an intuitive and simpler data flow control, most of the previous works use SDF instead of MDC for complexity concern. However, for MIMO FFT/IFFT, we found that if the data streams are properly scheduled, the utilization rate can increase from 25% to 100%. This makes MDC very suitable for MIMO-OFDM systems.

Therefore we propose an efficient mechanism of memory scheduling to reduce the required memory. Together with the proposed memory scheduling, the proposed MIMO MDC FFT/IFFT has the following advantages. First, the proposed memory scheduling mechanism reduces the size of storage elements. Moreover, the mechanism properly shuffles the four input streams such that stage one to stage five are all with the same feed-forward switch-box data flow. Therefore, the control simplicity of MDC schemes can be preserved while the memory size is greatly reduced. As for the utilization rate of butterflies and multipliers, each one of the four input symbols after memory scheduling takes 25% of one symbol-time for radix-4 butterfly computation. Consequently one radix-4 butterfly and three twiddle-factor multipliers in each pipeline stage can process four data streams without any idle period, that is, the utilization rate of butterflies and multipliers is 100%. Furthermore, the radix-8 butterfly at the last stage can be configured as a radix-4 butterfly. With such flexibility, radix-2 computation can be incorporated at the last radix-8 stage, and thus for any N in power-of-2 fashion can be computed with this proposed method. Finally, the serial blocks of output symbol format helps to reduce the memory usage for output sorting and the complexity of the modules followed by the FFT/IFFT processor.

For description convenience, the following notations are applied: i stands for spatial stream index, j stands for OFDM symbol index, n stands for input sample index, and k stands for output sample index. Thus each input sample can be represented as $x_j^i[n]$. Moreover, s denotes the pipeline stage, ranging from one to five in the proposed design. Fig. 3 shows the block diagram of the proposed MIMO FFT/IFFT computing core with $N = 2048$. The input order and the indices in between are also annotated.

A. Input Memory Scheduling

The goal is to convert the input streams in Fig. 3(a) to the format in Fig. 3(b). There are 12 memory banks at the input stage for converting the parallel input streams into serial blocks, such that one butterfly at each stage can compute the four data streams without idle period.

The 12 memory banks are grouped into four memory sets as shown in Fig. 4(a), that is, memory sets a, b, c, and d, which are used to store the input streams A, B, C, and D, respectively. There are two kinds of grouping methods, namely grouping for even indexed symbols and grouping for odd indexed symbols. Let the index of OFDM symbol begin from 0. For even-indexed OFDM symbols, the grouping method in the left side of Fig. 4(a) is used and for odd-indexed OFDM symbols, the grouping method in the right side of Fig. 4(e) is used. Fig. 4 illustrates the memory scheduling for even-indexed OFDM symbols. The scheduling for odd-indexed OFDM symbols will become clear after the illustration for even-indexed OFDM symbols. Let us take $N = 2048$ as an example and explain the input scheduling as follows.

Initially the 12 memory banks are logically grouped into four sets $\{a_1, a_2, a_3\}$, $\{b_1, b_2, b_3\}$, $\{c_1, c_2, c_3\}$, and $\{d_1, d_2, d_3\}$ as shown in Fig. 4(a). Each set is in charge of one input stream. From the first to the $3N/4$th cycle, the memory banks keep the first to $3N/4$th samples of each input stream. For the case of $N = 2048$, the memory banks $\{a_1, a_2, a_3\}$, $\{b_1, b_2, b_3\}$, $\{c_1, c_2, c_3\}$, and $\{d_1, d_2, d_3\}$ store the samples 1th–512th, 513th–1024th, 1025th–1536th} of the first, the second, the third, and the fourth input streams, respectively.

From the $(3N/4 + 1)$th to the Nth cycle shown in Fig. 4(b), the radix-4 butterfly processes the read-out data from the

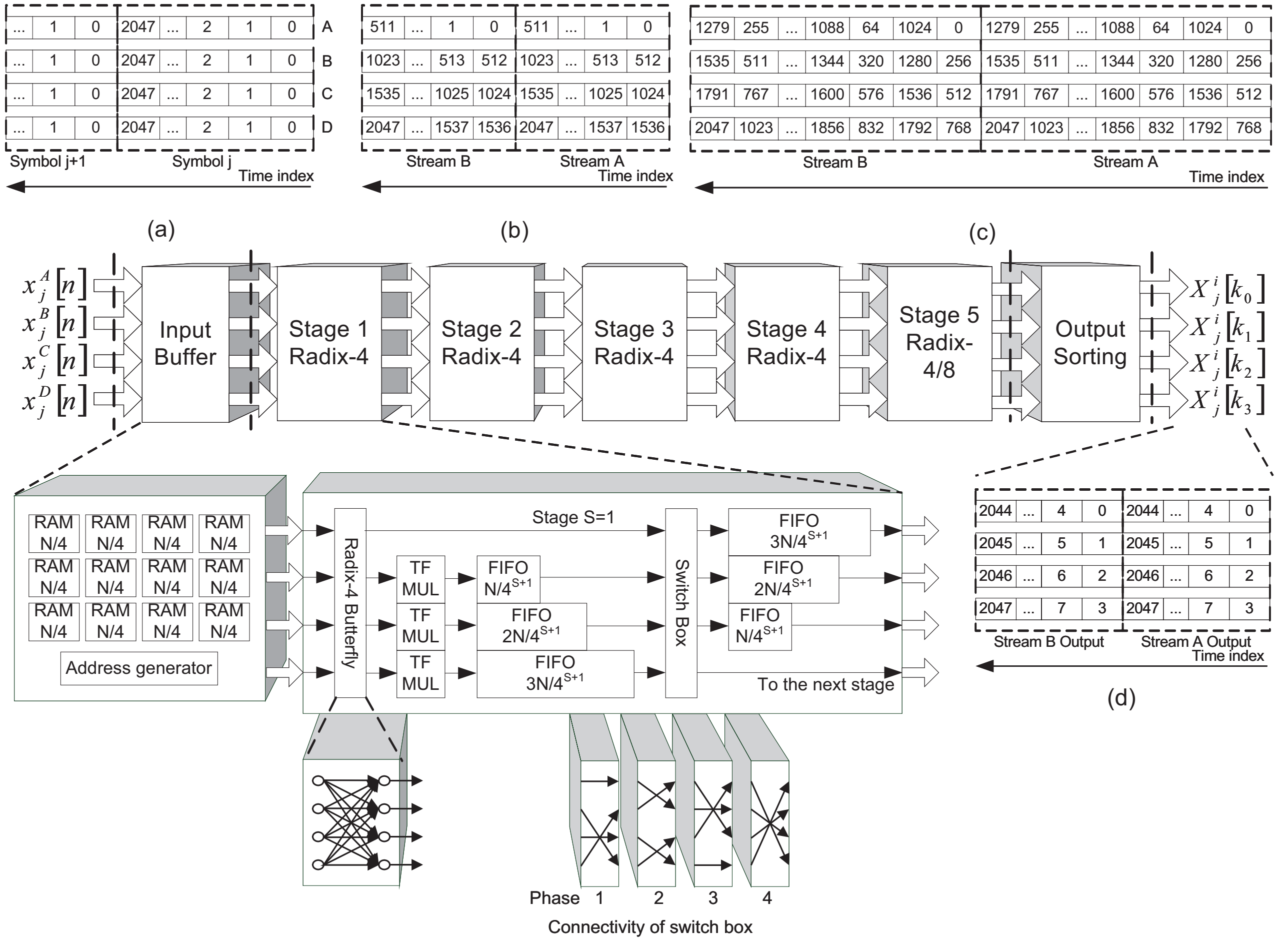


Fig. 3. Block diagram of the proposed MIMO MDC FFT/IFFT processor. The routing rule updates every $N/4^{s+1}$ clock cycles. (a) Initial input order. (b) Sorted input order at the output of input buffer. (c) Computed output order without sorting. (d) Output order after output sorting.

memory set $\{a_1, a_2, a_3\}$ and then this memory set are updated with the incoming samples from stream B, C, and D. That is, together with the previously stored first to $3N/4$th samples, now the radix-4 butterfly can process the samples of stream A, because the $(3N/4+1)$th to the Nth samples are ready at this moment, also, since only one butterfly is used at each stage, the $(3N/4+1)$th to the Nth samples for input streams B, C, and D are stored in the vacated memories a_1, a_2, and a_3, respectively. Continuing with the example of $N = 2048$, at the end of the 2048th clock cycle, the radix-4 butterfly has computed the 2048 samples of stream A, and the memory set $\{a_1, a_2, a_3\}$ is updated with the 1537th to the 2048th samples of stream B, C, and D, respectively.

Similarly, in the next $N/4$ cycles, the contents in memory set b are updated as shown in Fig. 4(c). The processor reads out the 2048 samples of stream B from the memory banks a_1 and $\{b_1, b_2, b_3\}$ and sends it to the radix-4 butterfly. Then, the empty memories a_1 and $\{b_1, b_2, b_3\}$ are updated by the first to the $N/4$th samples of streams A, B, C and D, respectively, of the second OFDM symbols. Continuing with the example of $N = 2048$, at the end of the 2560th clock cycle, the radix-4 butterfly has computed the 2048 samples of stream B, and the memories a_1 and $\{b_1, b_2, b_3\}$ are updated with the first to the 512th samples of stream A, B, C, and D, respectively, of the second OFDM symbols.

Similar procedure is executed for stream C and D, and this is shown in Fig. 4(d) and (e). Note that in Fig. 4(e), the memory grouping of b, c, and d are transposed logically while the grouping for a remains the same at the end of the $7N/4$th cycle, when compared to Fig. 4(a). Also, from Fig. 4(e), now the 12 memory banks already store the first to the $3N/4$th samples of the second OFDM symbol. Continuing with the example of $N = 2048$, at the end of the 3072th and the 3584th clock cycles, the radix-4 butterfly has handled streams C and D, respectively. Moreover, at the end of the 3584th clock cycle, all the memories are updated with the first to the 1536th samples of the second OFDM symbol. Next, similar procedures mentioned above are used to handle the second OFDM symbol. For a practical implementation, the control mechanism of the proposed input scheduling is summarized in Fig. 5, where the switch-box at stage s updates the routing rule every $N/4^{s+1}$ OFDM symbol time.

By using the proposed memory scheduling illustrated in Figs. 4 and 5, the input sequence in Fig. 3(a) is converted into the format shown in Fig. 3(b). In Fig. 3(b), each of the four scheduled sequences occupies $1/4$ of one OFDM

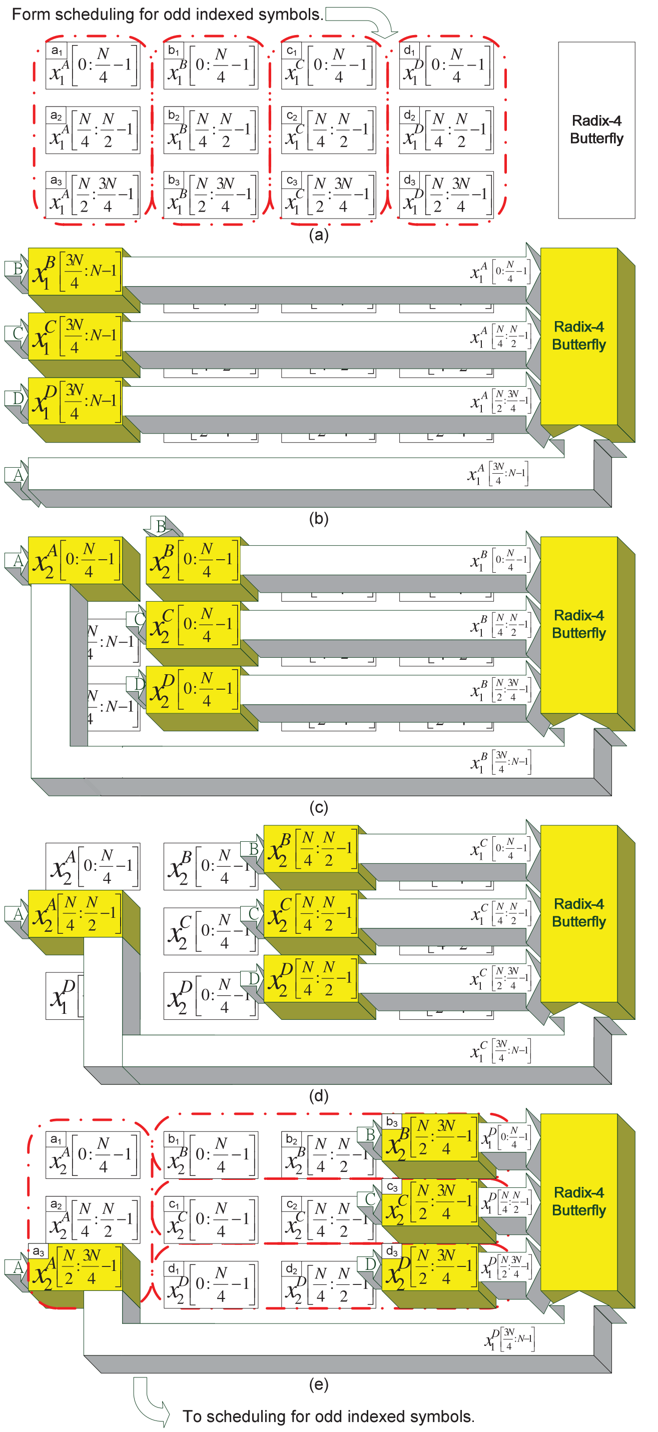
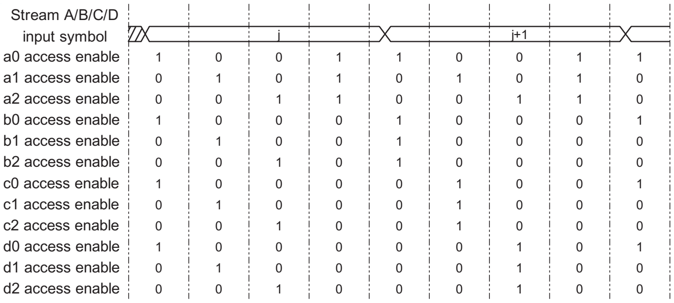

Form scheduling for odd indexed symbols.
Radix-4 Butterfly
(a)
(b)
(c)
(d)
(e)
To scheduling for odd indexed symbols.

Fig. 4. Illustration of the proposed input scheduling. The cubicles are the physical memory and radix-4 butterfly in active mode, while the rectangles are the modules in dormant mode. (a) Logical groups of initial memory banks. (b) $(3N/4+1)$th to the Nth cycle. (c) $(N+1)$th to the $(N+N/4)$th cycle. (d) $(N+N/4+1)$th to the $(N+N/2)$th cycle. (e) $(N+N/2+1)$th to the $(N+3N/4)$th cycle.

symbol time, hence all four scheduled sequences can be handled within one OFDM symbol duration using one radix-4 butterfly at each stage. As a result, the utilization rates for adders, multipliers and memories are 100%. The computational complexity for each stage is thus one radix-4 butterfly, three twiddle-factor multipliers, and a switch-box with first-in first-outs (FIFOs). Since stage s needs $3N/4^s$ words of FIFOs, together with the input scheduling memory that is of $3N$ words, the overall required memory size of the proposed radix-4 MDC FFT/IFFT processor with four parallel input streams is $3N + \sum_{s=1}^{\lfloor \log_4 N \rfloor - 1} 3N/4^s$.



Stream A/B/C/D input symbol			j				j+1		
a0 access enable	1	0	0	1	1	0	0	1	1
a1 access enable	0	1	0	1	0	1	0	1	0
a2 access enable	0	0	1	1	0	0	1	1	0
b0 access enable	1	0	0	0	1	0	0	0	1
b1 access enable	0	1	0	0	1	0	0	0	0
b2 access enable	0	0	1	0	1	0	0	0	0
c0 access enable	1	0	0	0	0	1	0	0	1
c1 access enable	0	1	0	0	0	1	0	0	0
c2 access enable	0	0	1	0	0	1	0	0	0
d0 access enable	1	0	0	0	0	0	1	0	1
d1 access enable	0	1	0	0	0	0	1	0	0
d2 access enable	0	0	1	0	0	0	1	0	0

Fig. 5. Memory access control of the proposed input memory scheduling. Each memory access performs write-after-read.

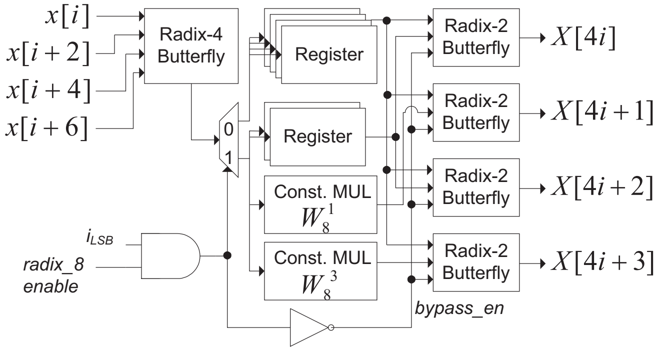


Fig. 6. Schematic of the proposed radix-8 butterfly. In radix-8 operation i is configured as 0 or 1, while in radix-4 operation i is 0.

B. Butterfly Operations

The proposed FFT/IFFT processor uses radix-4 butterflies as fundamental computing elements. Each stage adopts the same radix-4 butterfly, while the last stage uses a radix-8 butterfly which can also be configured as a radix-4 butterfly. As for the storage requirement of the twiddle factors, Lin suggested to keep only the twiddle factors whose phase indices are within $N/8$ [12], the rest of the twiddle factors can be derived from quadrant conversion. As for the complex multiplications, each radix-4 butterfly needs three multipliers and five real adders [13]. We adopted the routing rule for switch-box proposed by Swartzlander in [23].

We propose a configurable radix-8/radix-4 butterfly for the last stage, where the multiplications of twiddle factor can be realized by constant multipliers. This butterfly is composed by one radix-4 and four radix-2 butterflies as shown in Fig. 6. When a radix-4 instead of a radix-8 computation is needed, this butterfly enables only the internal radix-4 computations and disables the other radix-2 computations.

C. Memory Reduced Bit/Set Reversing Method

For an FFT/IFFT processor with radix-4 butterflies, the input and output indices are in set-reversed order instead of bit-reversed order. From the example in Fig. 3, if N is with power

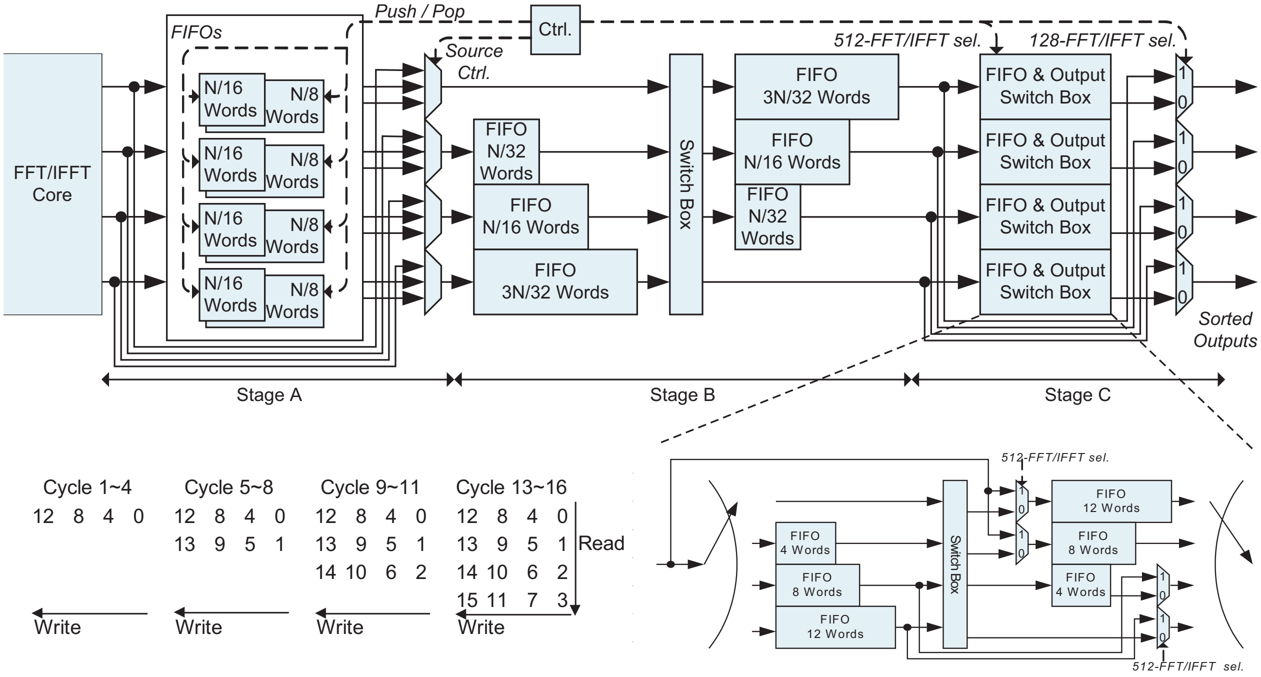


Fig. 7. Proposed output sorting for set-reversed FFT/IFFT processors. Stage A percolates the first half symbol from interlaced sequence. Stages B and C contain switch-boxes for reordering. The annotated write/read-access indices in stage C is used for 512-FFT/IFFT.

of 4, e.g., $N = 1024$, the last stage is configured as a radix-4 butterfly. Let the bit-wise input sample index n be expressed as $[b_9\ b_8 \cdots b_2\ b_1\ b_0]$, then the bit-wise output indices of the proposed MDC FFT/IFFT processor for k_1, k_3, k_5, k_7, and k_8 are $[b_9\ b_8]$, $[b_7\ b_6]$, $[b_5\ b_4]$, $[b_3\ b_2]$, and $[b_1\ b_0]$, respectively. On the other hand, if N is not with power of 4, e.g., $N = 2048$, the bit-wise output indices for k_1, k_3, k_5, k_7, and k_8 are $[b_{10}\ b_9]$, $[b_8\ b_7]$, $[b_6\ b_5]$, $[b_4\ b_3]$, and $[b_2\ b_1\ b_0]$, respectively. Since the memory dominates the area of an FFT/IFFT processor, we would like the memory requirement for bit/set-reversing be as small as possible. Based on the proposed MDC MIMO FFT/IFFT processor, we propose an output set-reversing method that only needs $9N/8+192$ words of memory while can handle four N-point data streams.

The proposed output sorting is to convert the output indices in Fig. 3(c) to those in Fig. 3(d). The output format in Fig. 3(d) reduces the hardware complexity due to the following reasons. First, the sequential outputs are straightforward for serial processing, such as cyclic-prefix insertion at the transmitter, and sub-channel equalization or frame reconstruction at the receiver. Second, for MIMO applications no duplicated macros are required. The baseband modules, which follows the FFT/IFFT processor can have full utilization rate and the complexity for system integration is therefore simplified. Finally, the required memory for output set-reversing is small.

The proposed architecture for output sorting is shown in Fig. 7. Let us explain how it works. The output indices in Fig. 3(c) shows that the first half, from indices 0 to $N/2-1$, and the rest of the output samples, which are from indices $N/2$ to $N-1$, are interlaced. Thus, the first step is to extract the first half of the samples. Note that the interlacing occurs in 2048, 512, and 128-point FFT/IFFT, where the radix-8 computation is needed at the last stage. For $N = 1024$, that is, power-of-4, the radix-8 computation is not required, and the output of the FFT/IFFT processor are not interlaced. The signal *Source Ctrl.* in Fig. 7 is used to select the first half or the other half of the OFDM symbol, or bypass the selection when $N = 1024$. The six FIFOs before and after the switch-boxes save the sequence shuffled by the radix-4 computation in FFT/IFFT core. Thus for $N = 2048$ and $N = 1024$, stages B and C in Fig. 7 are needed. For $N = 128$, stage C is bypassed. For $N = 512$, stage B is needed but stage C is only partially active, that is, in stage C the switch-box is disabled, and the FIFOs with sizes eight and twelve are applied for the sorting with the write/read-access indices shown in Fig. 7. Using the proposed output sorting, we need $3N/4$ words of memory to buffer interlaced sequence and $3N/8 + 192$ words of memory for switch-boxs. The overall required memory size is therefore $9N/8 + 192$.

Comparison With Previous Works: Let us compare the output bit/set reversing (reordering) method to the conventional methods herein. We first briefly describe the features of the conventional methods, and then compare the proposed method with them. The sorting method proposed by Fu [20] generates a set of 64 sequential output samples in 16 clocks. Let A_1, A_2, A_3, and A_4 be four consecutive 128-point FFT/IFFT symbols; also, let the first 64 samples and the last 64 samples in each symbol be denoted by A_{i1} and A_{i2}, respectively, that is, $[A_{11}\ A_{12}]$ for the first symbol, $[A_{21}\ A_{22}]$ for the second symbol, and so on. The output order of this sorting method will be $[A_{11}\ A_{21}\ A_{31}\ A_{41}\ A_{12}\ A_{22}\ A_{32}\ A_{42}]$, that is, the output is "interlaced." In [17], Parhi uses a life-time analysis to find out the timing relationship between inputs and outputs. This method can guarantee the minimum memory usage. However, this method needs several switches for the flow control of registers according to the permutation rules. Hence, the control mechanism is somewhat complicated and so is the corresponding routing. In this case, we may not be able to use a high-density memory. Also, the routing complexity increases in MIMO systems. Fan in [21] proposed an output sorting function for FFT/IFFT used in ultrawideband systems. To generate sequential output samples, two different memory access rules were applied for even and odd symbols. In a 128-point FFT/IFFT processor, the authors mentioned that this method can reduce the output sorting buffer from 320 to 128. To avoid access conflict, however, each output symbol is spaced with 9 to 10 clock cycles. As a result, the memory usage does not achieve the 100% utilization. In [18], Jarvinen *et al.* use iterative tensor product to describe the stride permutation in FFT sorting. Also, they proposed to use multiple small FIFOs to shuffle the inputs into desired output sequence for specific configuration. However, small FIFOs lead to complicated routing and large area when FFT/IFFT size is large. To overcome these issues, in [19], Puschel *et al.* proposed a clever reordering method, which uses two independent size-N RAMs to handle the output reordering. Since RAM macro is of high density, this method can significantly reduce the routing complexity and area. Also, the required memory size for output reordering is fixed to be $2N$ for multiple data streams. In the provided example, the stream number can be up to 11.

Since our design example is for LTE and Wi-MAX, the proposed output reordering scheme simultaneously supports various FFT/IFFT sizes, that is, $N = 128$, 512, 1024, and 2048, as well as multiple data streams. Due to the use of MDC architecture, the control mechanism for the output reordering is simple because the switch boxes at all stages have the same architecture, except that their operational clock counts are different. As a result, the memory control only demands push-and-pop operation instead of memory addressing, and the corresponding routing complexity is much simpler than that in [17] and [18]. Also, the proposed reordering enables the output samples to be consecutive, which is not like the "interlaced output" as in [20], and non-fully utilization of memory in [21]. Furthermore, compared to [19], the proposed reordering requires less memory than that in [19]. Taking $N = 2048$ for instance, the proposed method needs $(9/8)N + 192 = 2496$ words of memory while the method proposed in [19] needs $2N = 4096$ words of memory. However, it is worth emphasizing that the required memory size of the method proposed in [19] is always $2N$ for arbitrary N. On the other hand, since our proposed reordering considers an N with its maximum value be 2048 for the specific design applications, if N increases, the memory size may not be fixed as $(9/8)N + 192$, and needs to be re-evaluated in this case.

D. Reduction in Computing Elements

Let us compare the required adders and multipliers for the proposed MIMO MDC FFT/IFFT architecture and conventional schemes. For the conventional schemes, most works used radix-2 butterfly to implement the FFT/IFFT processor, e.g., [10], [12], [14]. For these schemes, there are $\log_2 N$ stages and each stage needs two complex adders and one complex multiplier. In MIMO systems with N_s data streams, the number of complex adder N_a and the number of complex multiplier N_m for conventional schemes are given respectively by

$$N_a = N_s \times 2 \times \log_2 N \tag{5}$$

and

$$N_m = N_s \times (\log_2 N - 1). \tag{6}$$

Note that in the last folding stage, all the twiddle factors are one. Therefore, multipliers are not required in the last pipe. For the proposed FFT/IFFT scheme, the discussion in previous sections are dedicated for practical MIMO systems with four data streams. Thus radix-4 butterflies are used. It is worthwhile to emphasize, however, that similar concept can be used for MIMO systems with arbitrary number of data streams. That is, for a system with N_s data stream, we propose to use radix-N_s butterfly with MDC architecture. In this case, there are $\log_{N_s} N$ stages. Also, thanks to the 100% utilization rate of MDC, each stage requires $N_s - 1$ multipliers, and $\sum_{k=0}^{\log_2 N_s - 1} 2^k$ complex adders if binary-tree addition is used in each radix-r branch. As a result, the number of complex adders N'_a and the number of complex multipliers N'_m are given respectively by

$$N'_a = \log_{N_s} N \times r \times \sum_{k=0}^{\log_2 N_s - 1} 2^k \tag{7}$$

and

$$N'_m = (r - 1) \times (\log_{N_s} N - 1). \tag{8}$$

Although there are other advanced hardware schemes, such as R4MDC or R2^2SDF as specified in [10], which may lead to different number of adders and multipliers, if a specific folding scheme is chosen, the numbers of radix-r butterflies and complex multipliers does not make much difference. Therefore, we tend to take radix-r butterfly and complex multiplier as fundamental building blocks for comparison. The number of complex adders and multipliers for different radix schemes are compared in Table I, which is referenced from Fu's study [20]. R2^2SDF is of the minimum hardware requirement in single-input single-output (SISO) the FFT/IFFT processor. An intuitive approach to extend SISO scheme into MIMO scheme is duplicating these computing elements according to the number of data streams.

To give an insight into the computational reduction of the proposed scheme, we compare the required numbers of adders and multipliers in Fig. 8. From Fig. 8(a), we see that the proposed scheme generally requires more adders than those in conventional schemes except for $N_s \leq 4$. It is observed in Fig. 8(b), however, the proposed scheme requires fewer multipliers than those in conventional schemes. Since the hardware complexity of a multiplier is usually much higher than that of a adder, the proposed scheme enjoys implementational advantages. From this figure, we see that for $N_s = 4$, which is our design example for LTE and Wi-MAX applications, the proposed radix-4 MDC architecture only need 12 complex multipliers while the radix-2^2 scheme and radix-2 scheme need at least 16 and 36 complex multipliers, respectively.

TABLE I
HARDWARE COMPLEXITY OF DIFFERENT FFT/IFFT ARCHITECTURES

	Scheme	CPLX. MUL #	Radix-r BF #	Memory size
SISO	R2MDC	$(\log_2 N - 1)$	$\log_2 N$	$3N/2 - 2$
	R2SDF	$(\log_2 N - 1)$	$\log_2 N$	$N - 1$
	R4SDF	$(\log_4 N - 1)$	$\log_4 N$	$N - 1$
	R4MDC	$3(\log_4 N - 1)$	$\log_4 N$	$5N/2 - 4$
	R2^2SDF	$(\log_4 N - 1)$	$\log_4 N$	$N - 1$
MIMO	Prop. MIMO MDC	$(r - 1)(\log_{N_s} - 1)$	$\log_{N_s} N$	$(N_s - 1)N + \sum_{s=1}^{\lfloor \log_{N_s} N \rfloor - 1} \cdot (N_s - 1)N/N_s^s$

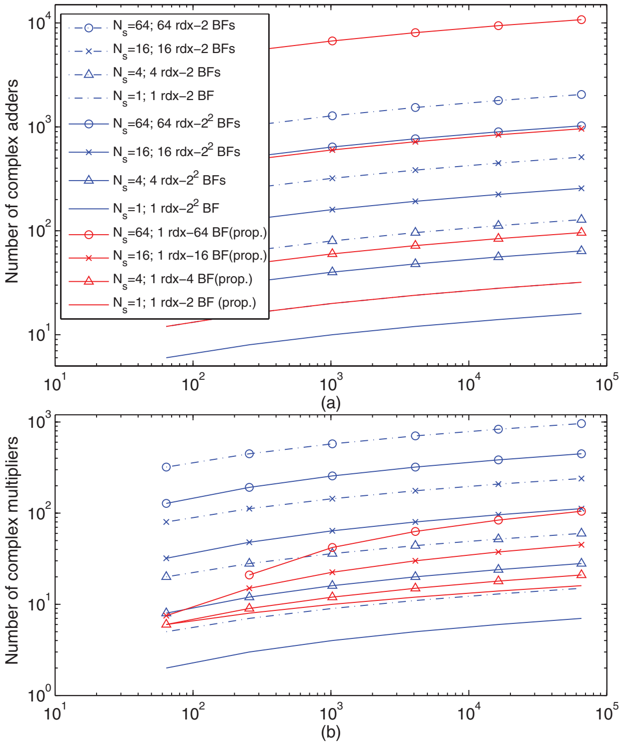


Fig. 8. Required number of (a) complex adders and (b) complex multipliers, as functions of FFT/IFFT size N for various numbers of MIMO data streams.

IV. IMPLEMENTATION

A. Hardware Specification and Synthesis Report

The required components of the proposed MDC MIMO FFT/IFFT processor are summarized in Table II. Lin *et al.* showed that using 12-bit internal word length can provide an output signal-to-noise ratio of 40 dB, capable of meeting the IEEE 802.16e WiMAX standard [31]. Based on out fix-point simulation, the input word length was fine tuned to 8 bits and the output word length was 12 bits. As for the internal word lengths, all the computations were rounded to 10 bits. The 12 memory banks, which store the input data were implemented by dual-port synchronous dynamic random access memory (SDRAM). The intermediate FIFOs whose depths exceed 8 were also implemented by dual-port SDRAM. The total SDRAM size in the proposed design was 23.56 KB.

TABLE II
ELEMENTS IN PROPOSED MDC FFT/IFFT PROCESSOR

Components	Purpose	Number
Complex number multipliers	Twiddle factors multiplication with radix-4/8 outputs	12
FFT butterflies	Radix-4	4
	Radix-4/8	1
Memory macros	Dual-Port SRAM (words)	10224
Other control modules	Switch-box	7
	Input memory addressing	1
	Output FIFO control	1
	FIFO registers (words)	456
	Quadrant conversion of twiddle factor	12
	Twiddle factor generator	4

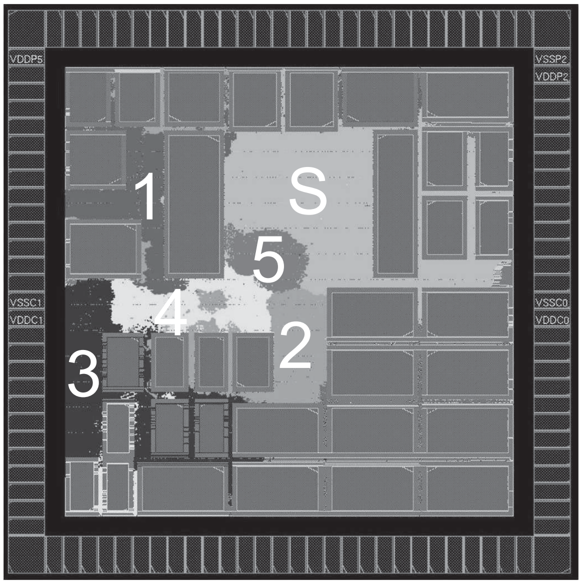
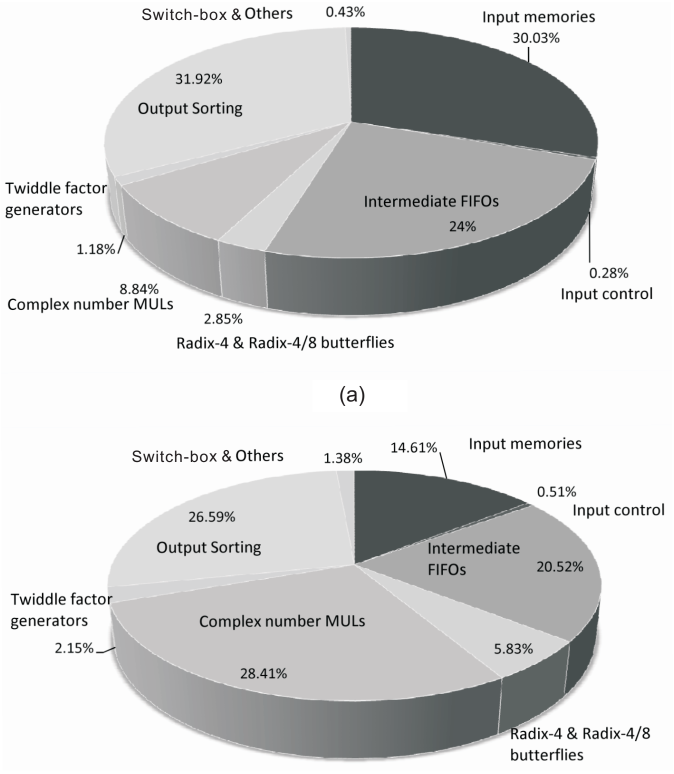


Fig. 9. APR of the proposed MDC MIMO FFT/IFFT processor. The pipeline stages of FFT/IFFT computing core are numerated from one to five. "S" is the output sorting stage. The memory control function is distributed in adjacent memory macros.

Other FIFOs with depths smaller than 8 were implemented by registers. The required twiddle factors in each stage can be implemented by table look-up using physical ROM macros.

The functionality of the proposed FFT/IFFT processor was implemented and verified by Cadence Verilog-XL simulation. The circuit was synthesized by Synopsys Design Compiler using an UMC 90-nm CMOS cell library. The system clock for synthesis was targeted at 40 MHz. It is worth pointing out that one of the advantages using pipeline architecture is the reduction of critical path. Pipeline registers were inserted at all outputs of memory macros, multipliers, radix-4 and radix-8 butterflies. In fact, we found the maximum achievable clock rate of the proposed design can be as high as 250 MHz in synthesis stage. The automatic place and route (APR) processing of the proposed FFT/IFFT processor was done by systems-on-a-chip (SoC) Encounter from Cadence. The core area was 3.1 mm^2. The APR result is shown in Fig. 9 with sub-block annotations.



(a)

(b)

Fig. 10. Statistics of memory and submodules. (a) Area. (b) Power.

The power consumption was analyzed primarily by Synopsys PrimePower with the net-list extracted from actual APR. We also use the power analysis function in SoC Encounter. The measured results using these two tools only have a small mismatch within 5 mW. Fig. 10 lists the pie diagram of area and power consumption. Excluding the testing function, the memory occupied 85.95% of the total area. The ratio of standard cells versus SDRAM macros was close to 1/5. This means if more advanced (that is, high density) memory macros were applied, further reduction could be achieved. The power consumption at 40 MHz system clock were 63.72 mW for 2048-FFT, 62.92 mW for 1024-FFT, 57.51 mW for 512-FFT, and 51.69 mW for 128-FFT computations.

B. Performance Analysis and Comparison

Throughput, signal to noise quantization ratio, and normalized area/power consumption are the major indices used to evaluate the performance of FFT/IFFT processors [22]. Let us compare the proposed design with other existing designs. Most of the previous works were based on SDF or memory-base radix-2^n algorithm. Bass proposed methods to evaluate the normalized area A_{bass} and normalized power consumption P_{bass} among different kinds of FFT processor [22] as follows:

$$A_{\text{bass}} = \frac{\text{Area}}{(\text{Tech}/0.5\ \mu\text{m})^2}$$

$$P_{\text{bass}} = \frac{\text{Tech} \times \left[\frac{2}{3}\frac{\text{Width}}{20} + \frac{1}{3}\left(\frac{\text{Width}}{20}\right)^2\right]}{\text{Power} \times \text{Exec Time} \times 10^{-6}}$$

where Tech is the process in micrometers, Width is the bit-width of data-path in bits, and ExecTime is the calculation time in microseconds. Baas's comparisons were used for FFT/IFFT designs that have the same N, and similar architecture with coarse adjustment corresponds to different CMOS process. In addition to the architecture, different FFT length N, system frequency, and applied CMOS processes fundamentally affect the area and power consumption. Thus Peng in [27] considered different FFT/IFFT length N and proposed the normalized area A_{peng} and normalized power P_{peng} as

$$A_{\text{peng}} = \frac{\text{Area}}{N \times (\text{Tech}/0.18)^2}$$

$$P_{\text{peng}} = \frac{\text{Power} \times \text{ExecTime}}{N \times (V_{DD}/1.8)^2}.$$

Now let us consider a more general evaluation as follows. The computational complexity for an N-point FFT/IFFT in radix-r is $N \log_r N$ [7]. In practical implementation, the addition and multiplication operations can be well scheduled and executed by a small number of radix-r butterflies and complex multipliers. In such a case, the complexity of different N-FFT/IFFT should grow in logarithmic scale instead of linear scale. As for different radix-r butterflies, the implementations are still based on fundamental radix-2 structure. When N increases, the number of pipeline stages is proportional to $\log_r N$. Therefore, the comparison should be normalized to the fundamental radix-2 structure. The power consumption is proportional to load capacitance, supply voltage, and operating frequency, that is, $P \propto CV^2F$. For comprehensive and comparable analysis among various N, architecture, technology and number M of data streams, we may revise the area metric as

$$A_{\text{propose}} = \frac{\text{Area} \times 10^3}{(\text{Tech}/0.09\ \mu\text{m})^2 \times M \times \log_2 N} \tag{9}$$

and the metric for power consumption as

$$P_{\text{propose}} = \frac{\text{Power} \times \text{ExecTime} \times 10^3}{M \times V_{DD}^2 \times N \log_2 N}. \tag{10}$$

Note that the V_{DD} in 0.09 μm process is 1 volt. There are still other factors that affect the comparing criterion, such as the type of applied RAM macros, the overall load capacitance, or different synthesis constraints. Meanwhile, the factor of system frequency is not included in the revised metrics. Generally different operating frequencies in similar design lead to different synthesis and APR results.

Table III compares the proposed scheme and other works. As previously stated, different fabrication technology and synthesis constraint affect the basis in comparison. Therefore, the FFT/IFFT processors with the same N are grouped for discussion. For FFT/IFFT processors with $N = 128$, the normalized-area for the MDC scheme in [20] is 67% $\sim$ 77% of that for SDF schemes in [12] and [14]. Note that with higher clock rate, the normalized energy is reduced at the cost of larger normalized area. The trend can be carried on to 512- and 2048-FFT/IFFT processors as that in [24] and [28]. Now consider the FFT/IFFT processors with a large size of $N = 2048$, where memory macros and storage elements dominate

TABLE III
COMPARISON AMONG DIFFERENT FFT/IFFT PROCESSORS

	Proposed				[26]	[27]	[29]	[31]	[30]	[24]	[28]	[25]	[20]	[14]	[12]
Architecture	MDC				SDF	MDF	Memory base	Memory base	SDF	SDF	Memory base	SDF	MDC	SDF	SDF
FFT size	2048	1024	512	128	128~2048	128~2048/1536	2048	2048	2048	2048	512	256	128	128	128
Clock rate (MHz)	40				40	35	20	22.86	45	300	324	300	75	40	250
Stream no.	4				1	1	4	2	1	1	1	8	4	4	4
Process (um)	0.09				0.18	0.18	0.18	0.13	0.35	0.09	0.09	0.09	0.18	0.13	0.18
Voltage (V)	1				1.8	1.8	1.8	1.2	3.3	1	1	0.85	1.8	1.2	1.8
Area (mm^2)	3.1				4.53	1.932	4.96	2.12	13.05	1.16	2.46	3.53	2.1	1.41	3.1
Output sorting	Yes				No	No	Yes	Yes	No	No	Yes	No	Yes	No	No
Power (mW)	63.72	62.92	57.51	51.69	55.64	11.29	52.02	17.26	640.00	159.00	103.50	119.70	87.00	5.20	175.00
Execute time (us)	51.20	25.60	12.80	3.20	51.25	234.00	205.2	89.76	22.36	0.85	0.22	0.11	0.43	0.8	0.13
Normalized energy	36.20	39.33	39.94	46.15	39.07	36.19	36.56	23.88	58.33	6.02	4.92	1.08	3.20	0.81	1.93
Normalized area	70.45				102.95	43.91	28.18	46.19	78.45	105.45	273.33	55.23	18.75	24.14	27.68

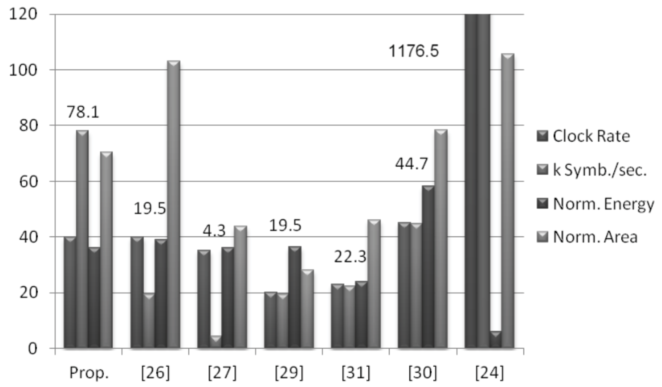


Fig. 11. Throughput comparison among various 2048-FFT/IFFT processors.

the die area. Although the normalized area in the memory-base processor is smaller, the normalized energy is not reduced proportionally [29], [31]. This is because the memory-based scheme uses twice amount of memory than those in pipeline schemes with continuous output. As long as the memories are accessed by computing elements, the macros consumes power. Consequently, to effectively reduce the area and power consumption in large N-FFT/IFFT processor, decreasing memory usage may be a key solution. Moreover, the output latency of the memory-based FFT/IFFT processors can be as long as one OFDM symbol. Thus it is not able to handle successive OFDM symbols unless extremely high clock is used to handle relatively slow data.

Trying to seek a good trade-off between these conventional schemes, the proposed FFT/IFFT processor adopts simple memory scheduling methods for both input and output data, this enables the processor to use a relatively small amount of memory to handle successive and multiple data streams. Observed from Fig. 10, the memory part, which includes input memories, intermediate FIFOs, and output sorting, takes 85.95% of the overall area and 61.72% of the overall power consumption. As a result, the scheduling methods not only reduce the area but also contribute to power saving. Comparing with [26], the proposed FFT/IFFT processor uses fewer computing elements, and the execution time is only one-forth of that for main distribution frame (MDF) scheme in [27]. Moreover, due to the use of output memory scheduling, the proposed FFT/IFFT processor can handle four data streams and produce bit/set-reversed output data simultaneously, from integration perspective, the adjacent functional blocks such as frequency domain equalizer can directly apply the bit/set-reversed results from FFT/IFFT processor without additional effort for reordering.

Fig. 11 converts the execute time per symbol into the throughput in terms of k-symbol per second. The corresponding clock rate, normalized power and area are also marked to show the design trade-off. Although the FFT/IFFT processor in [24] is of the highest throughput with the minimal normalized energy, it is at the cost of the largest normalized area and the maximal operational rate at 300 MHz. Note that the FFT in [24] was specifically for 16-quadrature amplitude modulation application and the word-size for real part and imaginary part is only 4-bit. Among the 2048-point FFT/IFFT processors with clock rate below 50 MHz, our proposed design is of the highest throughput.

V. CONCLUSION

In this paper, we proposed a radix-r based MDC MIMO FFT/IFFT processor for processing N_s streams of parallel inputs, where $r = N_s$ for achieving a 100% utilization rate. The proposed approach is suitable for MIMO-OFDM baseband processor such as WiMAX or LTE applications, where $N_s = 4$ and N can be configured as 2048, 512, 256, and 128. Moreover, we proposed an efficient memory scheduling to fully utilize memory. This considerably decreases the chip area because the memory requirement usually dominates the chip area in an FFT/IFFT processor. It is worth emphasizing that the proposed design is based on an MDC architecture, which is generally not preferred, due to its low utilization rate in memory and computational elements such as adders and multipliers. However, by using the proposed memory scheduling, MDC architecture is proved suitable for FFT/IFFT processors in MIMO-OFDM systems, because the butterflies and multipliers are capable of achieving a 100% utilization rate, meanwhile, the characteristics of simple control provided by MDC is maintained in the proposed design. The reduction in memory usage also leads to effective power saving, which is important for mobile devices. For applications applying large number N_s of data streams such as gigabit passive optical network, N_s can be as high as 64. In this case, the proposed radix-N_s MDC scheme and memory scheduling may also be applied to achieve a 100% utilization rate with simple control mechanism. Therefore, we conclude that the proposed designs found a good balance among complexity, energy consumption, and chip area, for the MIMO-OFDM systems.

Acknowledgment

The authors would like to thank the National Chip Implementation Center, Hsinchu, Taiwan, and the Industrial Technology Research Institute, Hsinchu, for chip implementation and technical support.

References

[1] *Asymmetric Digital Subscriber Line Transceivers 2 (ADSL2)*, ITU-T Standard G.992.3, Jan. 2005.

[2] *Very-High-Bit-Rate Digital Subscriber Line Transceiver 2(VDSL2)*, ITU-T Standard G.993.2, Feb. 2006.

[3] *The Wireless LAN Media Access Control (MAC) and Physical Layer (PHY) Specifications*, IEEE Standard 802.11, 1999.

[4] *IEEE Standard for Local and Metropolitan Area Networks. Part16: Air Interface for Fixed Broadband Wireless Access Systems*, IEEE Standard 802.16-2004, Oct. 2004.

[5] *IEEE Standard for Local and Metropolitan Area Networks. Part16: Air Interface for Fixed Broadband Wireless Access Systems*, IEEE Standard 802.16e-2005, Feb. 2006.

[6] Y. G. Li, J. H. Winters, and N. R. Sollenberger, "MIMO-OFDM for wireless communications: Signal detection with enhanced channel estimation," *IEEE Trans. Commun.*, vol. 50, no. 9, pp. 1471–1477, Sep. 2002.

[7] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1999.

[8] 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.

[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] S. He and M. Torkelson, "A new approach to pipeline FFT processor," in *Proc. IEEE Int. Parallel Process. Symp.*, Apr. 1996, pp. 766–770.

[11] A. Cortes, I. Velez, and J. F. Sevillano, "Radix r^k FFTs: Matricial representation and SDC/SDF pipeline implementation," *IEEE Trans. Signal Process.*, vol. 57, no. 7, pp. 2824–2839, Jul. 2009.

[12] 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.

[13] S.-H. Hsiao and W.-R. Shiue, "Design of low-cost and high-throughput linear arrays for DFT computations: Algorithms, architectures, and implementations," *IEEE Trans. Circuits Syst. II, Analog Digital Signal Process.*, vol. 47, no. 11, pp. 1188–1203, Nov. 2000.

[14] 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.

[15] Y. Jung, H. Yoon, and J. Kim, "New efficient FFT algorithm and pipeline implementation results for OFDM/DMT applications," *IEEE Trans. Consumer Electron.*, vol. 49, no. 1, pp. 14–20, Feb. 2003.

[16] T. Sansaloni, A. Perex-Pascual, V. Torres, and J. Valls, "Efficient pipeline FFT processors for WLAN MIMO-OFDM systems," *Electron. Lett.*, vol. 41, no. 19, pp. 1043–1044, Sep. 2005.

[17] K. K. Parhi, "Systematic synthesis of DSP data format converters using life-time analysis and forward-backward register allocation," *IEEE Trans. Circuits Syst. II, Analog Digital Process.*, vol. 39, no. 7, pp. 423–440, Jul. 1992.

[18] T. Jarvinen, P. Salmela, H. Sorokin, and J. Takala, "Stride permutation networks for array processors," in *Proc. 15th IEEE Int. Conf. Appl.-Spec. Syst., Archit. Process.*, Sep. 2004, pp. 376–386.

[19] M. Püschel, P. A. Milder, and J. C. Hoe, "Permuting streaming data using RAMs," *J. ACM*, vol. 56, no. 2, pp. 10:1–10:34, 2009.

[20] B. Fu and P. Ampadu, "An area efficient FFT/IFFT processor for MIMO-OFDM WLAN 802.11n," *J. Signal Process. Syst.*, vol. 56, no. 1, pp. 59–68, Jul. 2009.

[21] W. Fan and C.-S. Choy, "Robust, low-complexity, and energy efficient downlink baseband receiver design for MB-OFDM UWB system," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 59, no. 2, pp. 399–408, Feb. 2012.

[22] B. M. Baas, "A low-power, high-performance, 1024-point FFT processor," *IEEE J. Solid-State Circuits*, vol. 34, no. 3, pp. 380–387, Mar. 1999.

[23] E. E. Swartzlander, W. K. W. Young, and S. J. Joseph, "A radix 4 delay commutator for fast Fourier transform processor implementation," *IEEE J. Solid-State Circuits*, vol. 19, no. 5, pp. 702–709, Oct. 1984.

[24] 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, Exp. Briefs*, vol. 57, no. 6, pp. 451–455, Jun. 2010.

[25] Y. Chen, Y.-W. Lin, Y.-C. Tsao, and C.-Y. Lee, "A 2.4-Gsample/s DVFS FFT processor for MIMO OFDM communication systems," *IEEE J. Solid-State Circuits*, vol. 43, no. 5, pp. 1260–1273, May 2008.

[26] M. S. Patil, T. D. Chhatbar, and A. D. Darji, "An area efficient and low power implementation of 2048 point FFT/IFFT processor for mobile WiMAX," in *Proc. Int. Conf. Signal Process. Commun.*, 2010, pp. 1–4.

[27] S.-Y. Peng, K.-T. Shr, C.-M. Chen, and Y.-H. Huang, "Energy-efficient 128∼2048/1536-point FFT processor with resource block mapping for 3GPP-LTE system," in *Proc. Int. Conf. Green Circuits Syst.*, 2010, pp. 14–17.

[28] S.-J. Huang and S.-G. Chen, "A green FFT processor with 2.5-GS/s for IEEE 802.15.3c (WPANs)," in *Proc. Int. Conf. Green Circuits Syst.*, 2010, pp. 9–13.

[29] C.-L. Hung, S.-S. Long, and M.-T. Shiue, "A low power and variable-length FFT processor design for flexible MIMO OFDM systems," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2009, pp. 705–708.

[30] Y.-T. Lin, P.-Y. Tsai, and T.-D. Chiueh, "Low-power variable-length fast Fourier transform processor," *IEE Proc. Comput. Digital Tech.*, vol. 152, no. 4, pp. 499–506, Jul. 2005.

[31] Y. Chen, Y.-W. Lin, and C.-Y. Lee, "A block scaling FFT/IFFT processor for WiMAX applications," in *Proc. IEEE Asian Solid-State Circuits Conf.*, Nov. 2006, pp. 203–206.

Kai-Jiun Yang received the B.S. degree in electrical engineering from Tamkang University, Taipei, Taiwan, in 1999, and the M.S. degree in electrical engineering from the University of Southern California, Los Angeles, in 2001. He is currently pursuing the Ph.D. degree in electrical and control Engineering with National Chiao-Tung University, Hsinchu, Taiwan.

He was with Trendchip Technologies (merged into MediaTek Inc.), Hsinchu, from 2001 to 2009, and developed DMT-ADSL chip-set. He is currently with the Industrial Technology Research Institute, Hsinchu, where he participates in low-power wireless SoC implementation. His current research interests include baseband signal processing in orthogonal frequency division multiplexing systems, VLSI design, and verification.

Shang-Ho Tsai (S'04–M'06–SM'12) was born in Kaohsiung, Taiwan, in 1973. He received the Ph.D. degree in electrical engineering from the University of Southern California, Los Angeles, in 2005.

He was with the Silicon Integrated Systems Corporation, Hsinchu, Taiwan, from June 1999 to July 2002, where he participated in the VLSI design for DMT-ADSL systems. From 2005 to 2007, he was with MediaTek Inc., Hsinchu, and participated in the VLSI design for MIMO-OFDM systems. Since 2007, he has been with the Department of Electrical and Control Engineering (now Department of Electrical Engineering), National Chiao Tung University, Hsinchu, where he is an Associate Professor. His current research interests include signal processing for communication, statistical signal processing, and signal processing for VLSI designs.

Dr. Tsai received a Government Scholarship for overseas study from the Ministry of Education, Taiwan, from 2002 to 2005.

Gene C. H. Chuang (M'96) received the B.S. and M.S. degrees from National Chiao-Tung University, Hsinchu, Taiwan, in 1981 and 1983, respectively, and the Ph.D. degree from the Viterbi School of Engineering, University of Southern California, Los Angeles, in 1994.

He joined the National Telecommunication Laboratory, Yang-Mei, Taiwan, in 1984, and ITT/Qume separately. He then worked with IC Design Center, Taipei, Taiwan, and the System Laboratory of Philips Semiconductors, Taipei, for more than seven years. He was the Chief Architect and Co-Founder of Trumpion Microelectronics Inc., Taipei, from 1998 to 2004 and the Vice President of Cheertek Inc., Hsinchu, from 2005 to 2006. He is currently the Director of the Wireless Broadband Technology Division, Information and Communication Laboratory, Industrial Technology Research Institute, Hsinchu. He was involved in the integrated circuit design projects of LCD TV controller, satellite receiver, and multiple-input multiple-output WiMAX chip over the last ten years. His current research interests include signal processing and VLSI implementation, orthogonal frequency division multiplexing-based communication, and digital signal processing-based systems.