applied sciences

MDPI

Article

Area-Efficient FFT Kernel with Improved Use of GI for Multistandard MIMO-OFDM Applications

Song-Nien Tang [1,*] and Yuan-Ho Chen [2,*]

1 Information and Computer Engineering Department, Chung Yuan Christian University, Taoyuan 32023, Taiwan
2 Department of Electronics Engineering, Chang Gung University, and Institute for Radiological Research, Chang Gung University/Chang Gung Memorial Hospital, Taoyuan 33302, Taiwan
* Correspondence: sntang@ice.cycu.edu.tw (S.-N.T.); chenyh@mail.cgu.edu.tw (Y.-H.C.); Tel.: +886-3-265-4731 (S.-N.T.)

Received: 17 May 2019; Accepted: 16 July 2019; Published: 18 July 2019

Abstract: This study presents a fast Fourier transform (FFT) kernel for multistandard applications, which employ multiple-input, multiple-output orthogonal frequency-division multiplexing (MIMO-OFDM). The proposed design uses a mixed-radix, mixed-multipath delay-feedback (MRM^2DF) structure, which enables 4/5/6-stream 64/128-point FFT. This approach allows the effective usage of guard intervals (GI) in conjunction with a novel resource-sharing scheme to improve area efficiency. An area-reduced constant multiplication unit and sorting buffer with minimal memory size further reduced an area overhead. A test chip was designed using UMC 90-nm technology, and was evaluated through post-layout simulation. The proposed design outperformed previous works in terms of the throughput per area.

Keywords: fast Fourier transform; FFT; kernel; MIMO; OFDM; multistandard

1. Introduction

With the development of household and industrial applications, wireless data accesses between devices and users have become a concern on demand. Among various wireless communication schemes, multiple-input, multiple-output orthogonal frequency-division multiplexing (MIMO-OFDM) technology [1] are superior in transmission performances and are widely employed in numerous standards, such as WiFi (IEEE 802.11n/ac/ax), Long Term Evolution (LTE), and advanced 5G New Radio (NR) [2–5]. In practical applications, to access data from various devices, the user side must structure MIMO-OFDM communication according to the standard employed by the individual device. Therefore, developing a user-side transceiver, which can perform a multistandard MIMO-OFDM transmission for different devices, is useful (Figure 1).

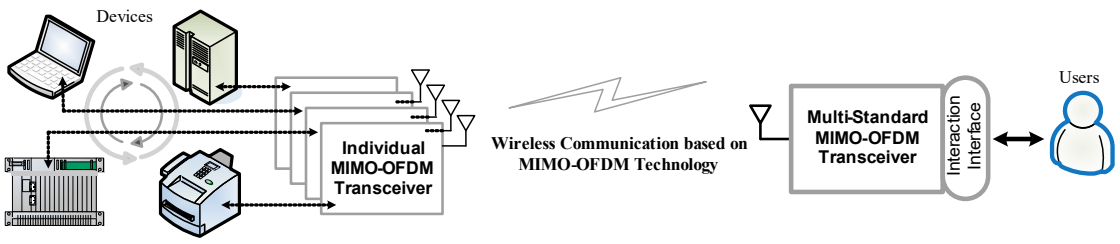


Figure 1. Scheme illustration of multistandard multiple-input, multiple-output orthogonal frequency-division multiplexing (MIMO-OFDM) data access applications.

For all MIMO-OFDM communication between users and devices, Figure 2 shows the block diagram of a conventional $M \times M$ MIMO-OFDM transmission model. A transmitter (TX) sends M streams of data symbols through M antennas. To reduce intersymbol interference (ISI) in OFDM communication, a guard interval (GI) was applied to the symbols by copying the number of samples from a symbol end to a symbol head. A receiver (RX) then uses M antennas to receive signals, which was obtained through the $M \times M$ MIMO channel. Figure 2 shows that the RX part contains M sets of radio frequencies (RF), analog-to-digital convertors (ADC), fast Fourier transforms (FFTs) and the following equalization and demodulation blocks. For the given N samples in an OFDM symbol, an $M \times M$ MIMO-OFDM receiver requires M-stream N-point FFT operations. The TX requires to perform M-stream N-point inverse FFT (IFFT) operations. Therefore, a set of M-stream N-point FFT (IFFT) processors are essential components in the MIMO-OFDM system. In general, the maximum M and N values for standard MIMO-OFDM FFT are 8 and 2048, respectively [6,7], and they can be higher in 5G applications [5]. Therefore, the cost of hardware and operation throughput for the required FFT/IFFT processors are significant design concerns. Further considering the FFT/IFFT design for multistandard MIMO-OFDM applications, implementing an effective scheme to develop the multimode hardware configuration is a difficult task. In this study, an FFT kernel design, which could be structured as a base module for reconfigurable FFT/IFFT processors, was proposed in a multistandard MIMO-OFDM system. The proposed FFT kernel could perform 4/5/6-stream 64/128-point FFT operations by efficiently using GI duration with resource-sharing and operation-rescheduling schemes. Therefore, the presented FFT kernel supports area-efficient and high-throughput development for multistandard MIMO-OFDM FFT/IFFT processors. The remainder of the paper is organized as follows. Section 2 outlines the design considerations for the FFT design and literature. Section 3 presents the hardware architecture of the proposed FFT kernel. Section 4 reports the implementation and comparison of the proposed FFT kernel design. Finally, conclusions are presented in Section 5.

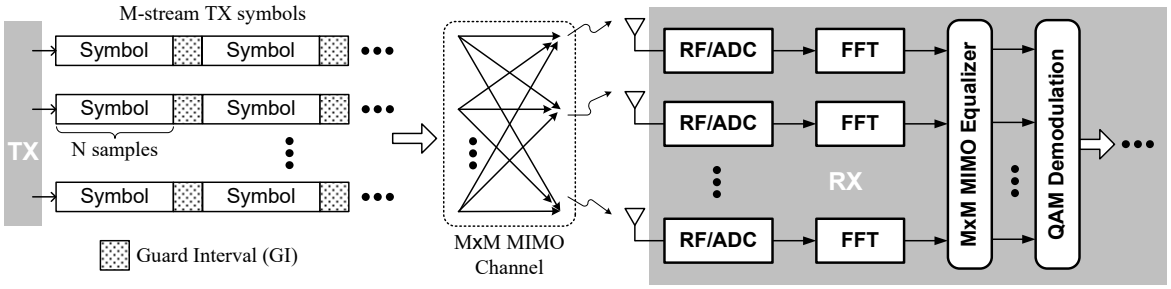


Figure 2. Block diagram for the conventional $M \times M$ MIMO-OFDM transmission model.

2. FFT Design Considerations

2.1. Related Works for MIMO-OFDM FFT

Various hardware architectures have been developed to perform the FFT algorithm. Among these architectures, the memory-based [8] and pipelined [9] architecture are the two primary categories. The memory-based FFT configuration is structured based on the employment of one or several processing elements (PEs) in cooperation with the memory modules. This memory-based structure is suitable for flexible FFT operations, which provide various FFT streams or operation points (i.e., FFT length) [10–12]. However, this approach is not appropriate for high-throughput or low-latency FFT operations when the operability of PEs or memory bandwidths is limited [13–15]. By using flexible FFT computation (length or streams), numerous memory-based FFT designs have been presented for MIMO-OFDM applications [15–17]. However, most of these approaches employ low-radix PEs (i.e., radix-r, where r is ≤ 16), which limits throughput performance. The pipelined FFT architecture is applied to increase operation throughput and latency by using hardware resources [9]. Moreover,

two conventional pipelined FFT categories are multipath delay communicator (MDC) [9,18] and single-path delay-feedback (SDF) architectures [9,19].

The MDC structure has the features of multipath (i.e., parallel) feedforward operations performed using switch control with first in first out (FIFO) memory. By effectively using the parallel data paths and operation units, the multipath processing feature of MDC can be applied to the FFT computation of multiple streams [20]. Therefore, several FFT designs based on MDC structures have been presented for MIMO-OFDM applications [20–23]. Figure 3 shows the block diagram for the conventional *M*-path MDC architecture applied to MIMO-OFDM FFT of *M* streams, where the feedforward multipath data is processed using switch blocks, FIFOs, butterfly units and multipliers. The SDF architecture provides a feedback path for FIFOs to efficiently manage the butterfly operation data at each pipeline stage [9]. To enable parallel processing for SDF configurations, the extended multipath delay-feedback (MDF) architecture [24,25] was proposed. Because the MDF scheme can process multiple input streams, MDF architectures have been researched for required multistream FFT operations in MIMO-OFDM systems [7,25–29]. On the basis of the MDF structure using the radix-2^k algorithm (where k is a positive integer) [30], an N-point FFT unit has $\log_2(N)$ radix-2 operation stages involving the feedback-pathed FIFO and butterfly 2 (BF2) units [9,30]. Figure 4 shows a conventional *M*-path MDF FFT structure for $M \times M$ MIMO-OFDM applications. Furthermore, some previous studies [21,25,28] discuss the schemes of hardware cost reduction for multipliers located at multiple paths (Figures 3 and 4). In general, these MDC or MDF approaches [20–29] enable multipath/parallel operations associated with multiple FFT streams (i.e., the number of data paths and FFT streams is equal).

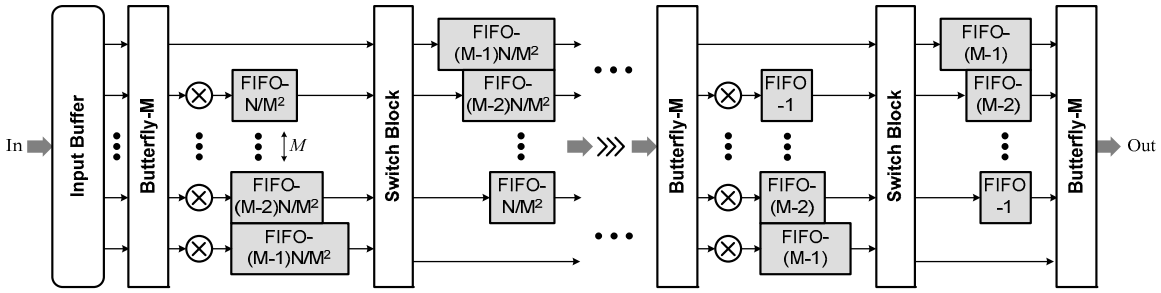


Figure 3. Block diagram of conventional *M*-path MDC fast Fourier transform (FFT) architecture for M-stream MIMO-OFDM.

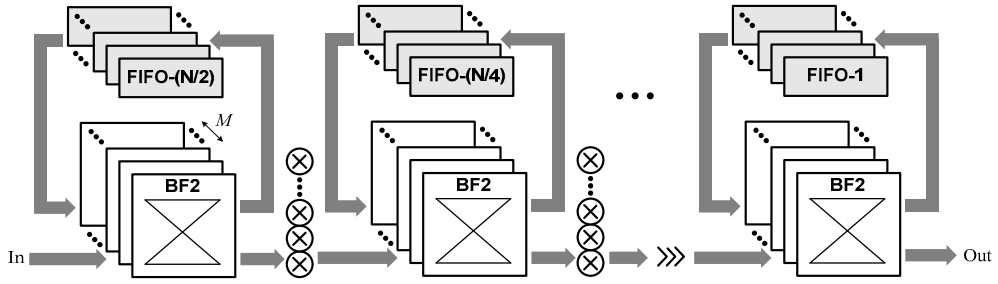


Figure 4. Block diagram of conventional *M*-path MDF FFT architecture for *M*-stream MIMO-OFDM.

2.2. Kernel-Based FFT for Multiple Standards

Most MIMO-OFDM standards have variable FFT lengths and stream levels, and thus, the corresponding MDC or MDF FFT designs generally employ a reconfigurable structure to support various FFT operation modes specified at the aimed specification [7,20,22]. In addition, considerable research has been conducted on advanced restructure schemes for multimode FFT processors that support multiple MIMO-OFDM standards [28,29]. For such multimode/multistandard MIMO-OFDM FFT applications, the aforementioned studies have employed tailored reconfiguration methods to enable multimode FFT operations based on their individual MDC/MDF structures. Therefore,

the aforementioned approaches lack the design flexibility and convenience to develop the multistandard MIMO-OFDM FFT architecture.

However, among the mainstream MIMO-OFDM standards (WiFi, LTE, and 5G applications), 1- to 4-stream 128/64-point FFT is implemented with a base operation mode [20–23,25–28]. This is one approach, using which a common optimized four-stream 128/64-point FFT kernel can be developed as a base module to construct specified *M*-stream *N*-point FFT/IFFT processors. For example, in the MDF FFT architecture, Figure 5 presents the aforementioned scheme. Considering *M*-stream *N*-point FFT computation ($M > 4$; $N > 128$, and N is to the power of 2), each *N*-point FFT can be performed using *N*′-point FFT in conjunction with 128-point FFT based on the radix-*N*′/128 FFT algorithm, where *N*′ is equal to *N*/128, and *M*-stream FFT can be implemented in $\lceil M/4 \rceil$ sets of four-stream FFT; $\lceil . \rceil$ denotes a ceiling operation. Figure 5 shows a four-stream 128/64-point FFT kernel prepared first as a common base module. This kernel module was further integrated with a front four-stream *N*′-point FFT computation unit to complete a set of four-stream *N*-point FFT. Such configuration (FFT kernel and *N*′-point FFT unit) can be extended to $\lceil M/4 \rceil$ sets, and complete *M*-stream *N*-point FFT operations can be implemented. For hardware modularization, kernel-based FFT configurations are more efficient and flexible when applied to scalable *M*-stream *N*-point FFTs for multiple MIMO-OFDM standards. Figure 5 presents a configuration which can be further modified and extended to integrate a conventional FFT kernel with MDC or memory-based units for target FFT computation. Considering four-path 128/64-point FFT kernel designs, employing the available FFT processors, which can perform the same FFT specification, is an efficient approach [21,25]. Nevertheless, for system optimization, specific development of a hardware-efficient FFT kernel as a common module can be advantageous. In this study, the target FFT kernel module was designed using a modified MDF structure as support. In addition to the original four-stream 128/64-point FFT operations, the proposed FFT kernel used GI duration to enable five- and six-stream FFT operations, and thereby enhance the overall throughput with an improved area efficiency. The concept for GI utilization for the FFT design has been mentioned in [31] for the purpose of improving the operation latency. In our design, we further utilized the GI in common with a resource-sharing scheme to increase the operation throughput. The details of the proposed design are discussed in Sections 3 and 4.

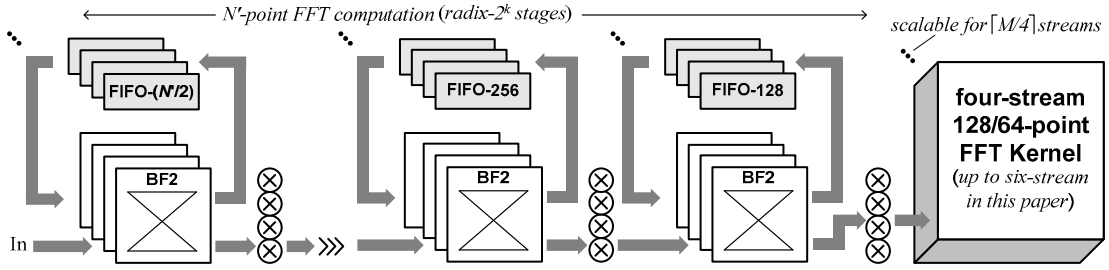


Figure 5. Block diagram showing the hardware architecture of M-stream N-point FFT processor using a. four-stream 128-point FFT kernel.

3. Proposed FFT Kernel Architecture

3.1. Algorithm and Architecture Overview

In conventional MDC/MDF FFT schemes [20–26], the number of paths is equal to the number of FFT streams. By contrast, a mixed-radix, mixed-multipath delay-feedback (MRM^{2}DF) architecture was developed, which operated on a radix-dependent number of paths. For 128-point FFT operations, the MRM^{2}DF design operated based on the mixed radix-2/8/8 algorithm, which can be described using Equations (1)–(5). A 128-point discrete Fourier transform (DFT) of a time domain sequence x(n), was defined as Equation (1), where X(k) and $W_{128}^{nk} = \exp(-j2\pi nk/128)$ are the DFT results and twiddle factor, respectively. Moreover, n and k are represented using Equation (2), and Equation (1) can be derived using Equation (3), a two-step computation based on radix-2 and radix-64 operations

represented as BF2 and 64-point DFT, respectively. Between BF2 and 64-point DFT operations, 128-based (W128) twiddle factors must be multiplied with the BF2 output. Furthermore, the 64-point DFT operation in Equation (3) can be further decomposed into the radix-8/8 operation.

By using Equation (4), a 64-point DFT can be derived as Equation (5), where two-stage radix-8 operations are performed. The two radix-8 stages correspond with the first and second butterfly 8 (BF8) operations, respectively. Similarly, 64-based (W64) twiddle factor multiplication is required between the first and second BF8 operations. For 64-point FFT operations, the MRM2DF design executes the radix-8/8 algorithm as Equation (5).

$$X(k) = \sum_{n=0}^{127} x(n)\, W_{128}^{nk},\ k = 0,\ 1,\ \ldots,\ 127 \tag{1}$$

$$\begin{cases} n = 64n_1 + n_2, n_1 = 0,\ 1; n_2 = 0,\ 1,\ \ldots,\ 63 \\ k = k_1 + 2k_2,\ k_1 = 0,\ 1; k_2 = 0,\ 1,\ \ldots,\ 63 \end{cases} \tag{2}$$

$$\begin{aligned} X(2k_2 + k_1) &= \sum_{n_2=0}^{63} \sum_{n_1=0}^{1} x(64n_1 + n_2) W_{128}^{(64n_1+n_2)(2k_2+k_1)} \\ &= \sum_{n_2=0}^{63} \left\{ \underbrace{\sum_{n_1=0}^{1} x(64n_1 + n_2)\, W_2^{n_1 k_1}}_{\text{BF2 (radix}-2\text{) operation}}\, W_{128}^{n_2 k_1} \right\} W_{64}^{n_2 k_2} = \underbrace{\sum_{n_2=0}^{63} \text{BF2}\,(k_1, n_2)\, W_{64}^{n_2 k_2}}_{\text{64}-\text{point DFT (radix}-64\text{)}} \end{aligned} \tag{3}$$

$$\begin{cases} n_2 = 8\alpha_1 + \alpha_2,\ \alpha_1\ \&\ \alpha_2 = 0,\ 1,\ \ldots,\ 7 \\ k_2 = \beta_1 + 8\beta_2,\ \beta_1\ \&\ \beta_2 = 0,\ 1,\ \ldots,\ 7 \end{cases} \tag{4}$$

$$\begin{aligned} X(2(\beta_1 + 8\beta_2) + k_1) &= \sum_{\alpha_2=0}^{7} \sum_{\alpha_1=0}^{7} \text{BF2}\,(k_1,\, 8\,\alpha_1 + \alpha_2)\, W_{64}^{(8\,\alpha_1+\alpha_2)(\beta_1+8\beta_2)} \\ &= \underbrace{\sum_{\alpha_2=0}^{7} \left\{ \underbrace{\left[\sum_{\alpha_1=0}^{7} \text{BF2}(k_1,\, 8\alpha_1 + \alpha_2)\, W_8^{\alpha_1 \beta_1} \right]}_{1^{\text{st}}\ \text{BF8 (radix}-8\text{) operation}}\, W_{64}^{\alpha_2 \beta_1} \right\} W_8^{\alpha_2 \beta_2}}_{2^{\text{nd}}\ \text{BF8 (radix}-8\text{) operation}} \end{aligned} \tag{5}$$

Corresponding with the radix-2/8/8 algorithm, the proposed MRM^{2}DF kernel was operated with hybrid path configurations (i.e., mixed-multipath). Hardware units associated with the radix-2 (BF2) computation allowed operations on at most six data paths (i.e., 4/5/6 paths) based on the number of streams. By contrast, radix-8 (BF8) hardware units allowed operations on eight paths corresponding to the number of radices. A shared hardware module was employed to perform W_{128} and W_{64} twiddle factor multiplication in six and eight paths, respectively. When performing 64-point FFT, the MRM^{2}DF kernel performed only eight-path operations based on the radix-8/8 algorithm. The features of the proposed MRM^{2}DF structure are as follows:

(i) In mixed-multipath operations based on streams or radix-8, GI duration can be employed to conduct 128-point FFT operations in up to six streams.
(ii) Resource sharing is applied to multipath radix-2/8/8 operations by using an area-reduced constant multiplication unit.
(iii) The required memory size (i.e., the number of stored data elements) are maintained at a modest level by using a sophisticated sorting scheme applied to the buffer.

(iv) Points (i)–(iii) allow significant improvements in area efficiency compared with available designs that support multipath 128-point FFT (e.g., [21,25]).

Figure 6a presents the block diagram of the proposed MRM^{2}DF FFT kernel, comprised of four modules (Modules 1–4). The operations associated with Modules 1–4 are illustrated in Figure 6b (association with color of Modules). Figure 6b shows that Module 1 is tasked with 4/5/6-path BF2 (radix-2) operations. Module 4 provides eight-path first and second BF8 (radix-8) computations.

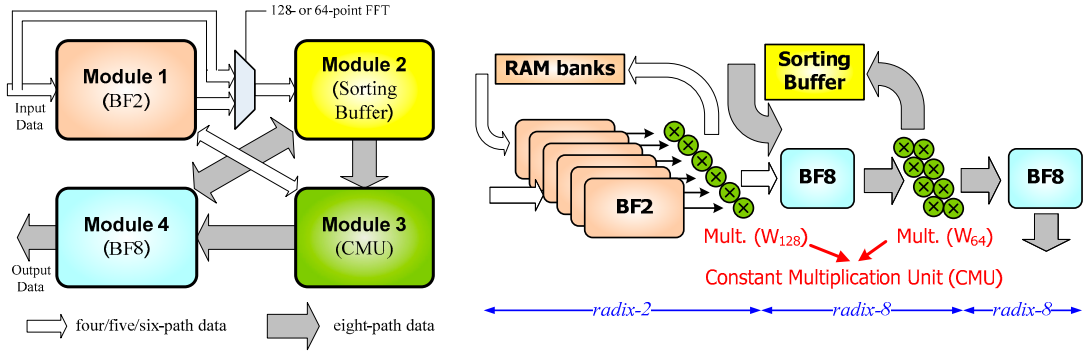


Figure 6. (**a**) Block diagram of proposed MRM^{2}DF FFT kernel; (**b**) module-associated operations.

Intermediate data transfer for radix-2/8/8 calculations was achieved using a sorting buffer in Module 2. As mentioned previously, W_{128} and W_{64} twiddle factors must be multiplied with the output of the radix-2 and first radix-8 stage. Module 3 was a constant multiplication unit (CMU) for performing twiddle factor multiplications by using a novel resource-sharing scheme. Modules 1–4 are detailed in Section 3. For 64-point FFT, only Modules 2 and 3 were used to perform radix-8/8 operations.

For example, for the six-stream 1024-point MDF FFT, Figure 7 shows an operation flow to generate a 128-point FFT block based on the MDF processing (Figure 5). In most OFDM standards (e.g., IEEE 802.11ac for WiFi), the GI period is 1/4 or 1/8 of the symbol duration, depending on the channel conditions. For example, six streams of 1024 data samples were accessed to perform 1024-point FFT operations, and the number of GI samples was 256 (i.e., 1/4 symbol duration). Figure 5 shows that this mechanism allowed the calculation of six-stream 1024-point FFT through the radix-2^3 stages and 128-point FFT block. The corresponding radix-2^3 signal flow chart with the radix-2 (BF2) operations is shown in Figure 7 for reference. Through the three stages of MDF-based radix-2 operations (including twiddle factor multiplication), sets of 512/256/128-point FFT sequences were generated (corresponding with the color and slash in the signal flow chart). Finally, eight sets of six-stream sequences were generated, including 128 data and 32 GI' samples. For each set of 128-sample data streams, the proposed MRM^{2}DF kernel was prepared to process six-stream 128-point FFT. Figure 8 presents the detailed timeliness diagram of MRM^{2}DF hardware operations and signal flow chart based on the radix-2/8/8 algorithm. The inclusion of GI duration in the time available for processing OFDM FFT is an efficient approach. However, for conventional MDC/MDF FFT structures operating at the input sample rate [20–22,25,26], the operations were generally idle during GI clock cycles, thereby reducing hardware efficiency. By contrast, the proposed MRM^{2}DF scheme used all 160 clock cycles, including GI' (Figure 8), for 128-point FFT operations.

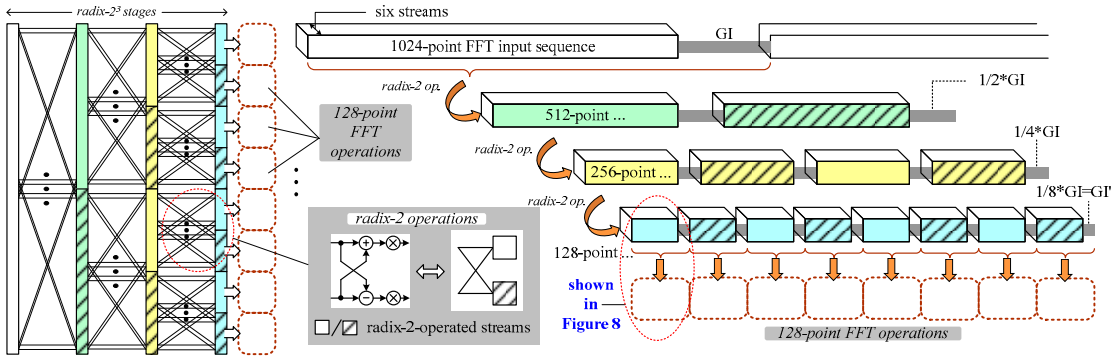


Figure 7. Operation flow for the generation of 128-point FFT block based on 1024-point MDF FFT.

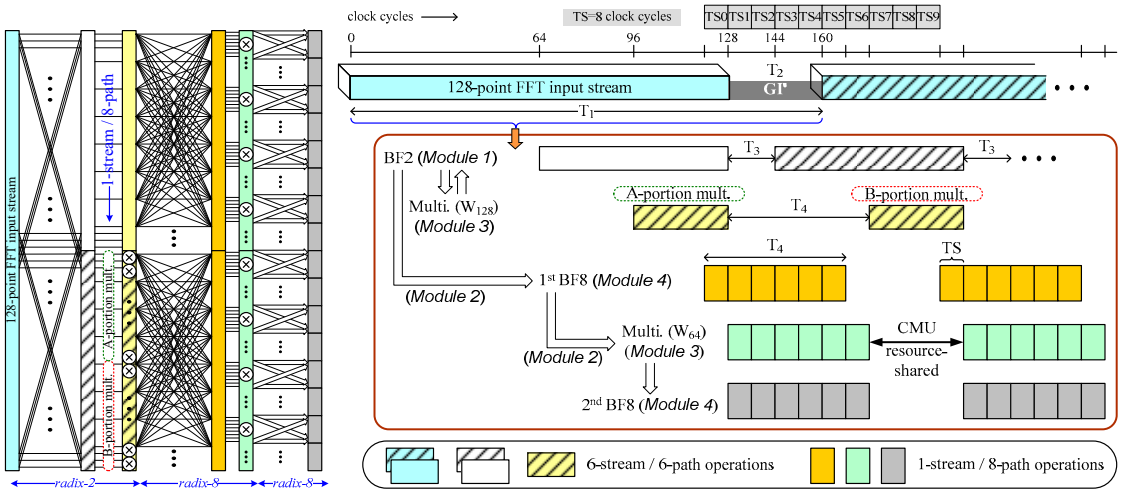


Figure 8. Detailed timeliness diagram for operations of the proposed MRM^{2}DF-structured FFT kernel and its associated signal flow chart for the radix-2/8/8 algorithm.

Figure 8 illustrates that BF2 and W_{128} multiplications were performed concurrently for all six data streams. BF8 and W_{64} multiplications were performed on eight stream-by-stream, based on a time unit of eight cycles (i.e., a time slot (TS)) for the operation of each stream. The radix-2/8/8 signal flow chart reveals that only half of the BF2 outputs (i.e., BF2 subtraction terms) were multiplied using W_{128} twiddle factors. This observation indicated that W_{128} multiplication was performed only during some of the 160 cycles (A- or B-portion mcl in green or red colors). This observation ensures that the non-occupied duration is available for W_{64} multiplication by using the resource-shared CMU. Moreover, using the TS unit (eight cycles) for BF8/W_{64} operations per stream allowed the efficient use of GI' for the processing of additional streams. In the aforementioned example, GI' had 32 cycles (i.e., T_2), and T_3 was half of T_2. Thus, six TS units (T_4) were available for BF8/W_{64} operations for six-stream 128-point FFT. Similar schemes can be applied to other GI' situations by evaluating T_1–T_4 terms in Figure 8. Table 1 lists T_1–T_4 parameters associated with various operating modes of 128-point FFT. In the 64-point FFT configuration (Figure 6), the proposed scheme provided only BF8/W_{64} operations based on the BF8/W_{64}-related portions of the timing schedule (Figure 8).

Table 1. Parameters for various operating modes.

GI' Clock Cycles	No. of Streams (max.)	T_1 Clock Cycles	T_2 Clock Cycles	T_3 Clock Cycles	T_4 Clock Cycles
32 (1/4 symbol)	6	160	32	16	48 (6 TS)
16 (1/8 symbol)	5	144	16	8	40 (5 TS)
0 (None)	4	128	0	0	32 (4 TS)

3.2. Architecture Modules

3.2.1. Module 1

The block diagram in Figure 9 shows that Module 1 comprises six sets of BF2 units and two banks of 32-element RAM modules for radix-2 operations. The path-routing control of multiplexers (M1–M5) allows Module 1 to perform six-path addition and the subtraction of the input for BF2 execution or route the read/write the data of six-set RAM banks for I/O data delivery. Module 1 operations were more complex than those conventionally used for SDF or MDF radix-2 stages [9,24–26] because this process enables data access for W_{128} multiplications by using the shared CMU (Figure 8).

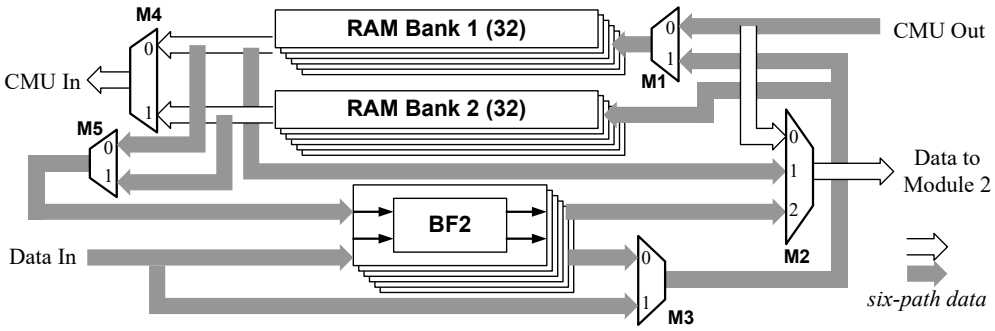
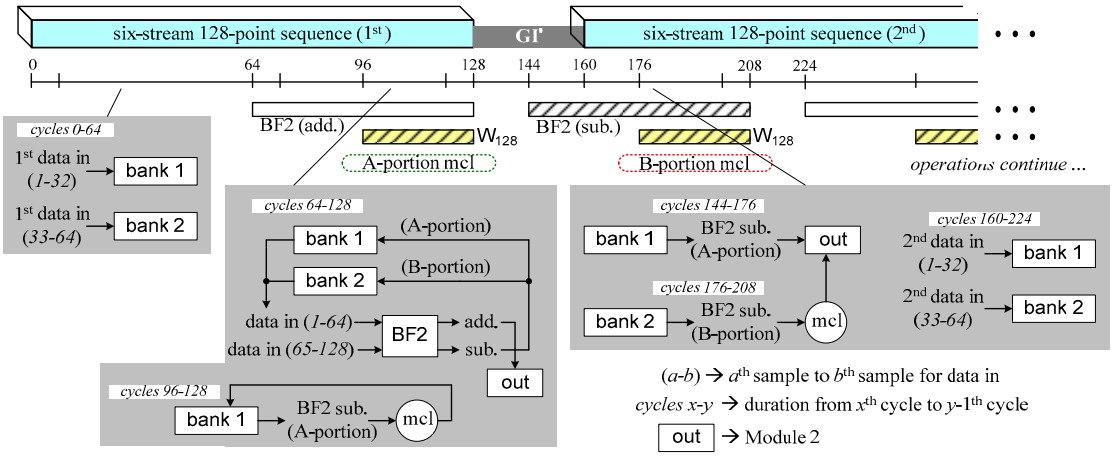


Figure 9. Block diagram of Module 1.

Referring to the top radix-2 related portions in Figure 8, Figure 10 details the operations of Module 1 and the controls of multiplexers M1–M5 for the first (the initial) and second 128-point input sequences of the six streams. Figure 10 presents addresses based on the schedule of clock cycles and radix-2 operations with W_{128} multiplication (Figure 8).



Mux \ cycles	0~32	32~64	64~96	96~128	128~144	144~160	160~176	176~208	208~224
M1	1	X	1	0	X	X	1		
M2	X		2		X	1		0	X
M3	1		0		X	1			
M4	X		X	0	X	X		1	X
M5	X		0	1	X	X		X: *don't care*	

Figure 10. Detailed operations of Module 1 and control of multiplexers (M1–M5) based on clock cycles.

3.2.2. Module 2

Figure 11 presents the block diagram of Module 2 (i.e., the sorting buffer), comprising six chains of shift registers (SRs) and multiplexers. The six streams of data from Module 1 were sent or routed to six SR chains (1–6) through paths A to F, respectively. Module 2 conducted the intermediate data

sorting of raidx-2/8/8 operations. For the radix-2/8 calculation, six-stream radix-2 data from Module 1 were stored in six-chain SR and then selectively read out to Module 4 for the first BF8 operations. The calculated results were stored back into the SR chain 1 by using configurable path routing and were prepared for subsequent radix-8/8 calculation. To continue radix-8/8 operations, eight data paths were selected from SR chain 1 and the output to Module 3 (CMU) for W_{64} multiplication and the subsequent execution of the second BF8. Figure 11 shows each SR chain was structured using eight SR units, which were classified into two types: Type I and Type II. Both types exhibited eight stages of flip flops and front multiplexers to support the data shift and hold. Type I was used for SR chain 1, whereas providing parallel data accesses for eight flip flop stages. Type II was used for SR chains 2–6 and optionally enabled the routing input from other SR units.

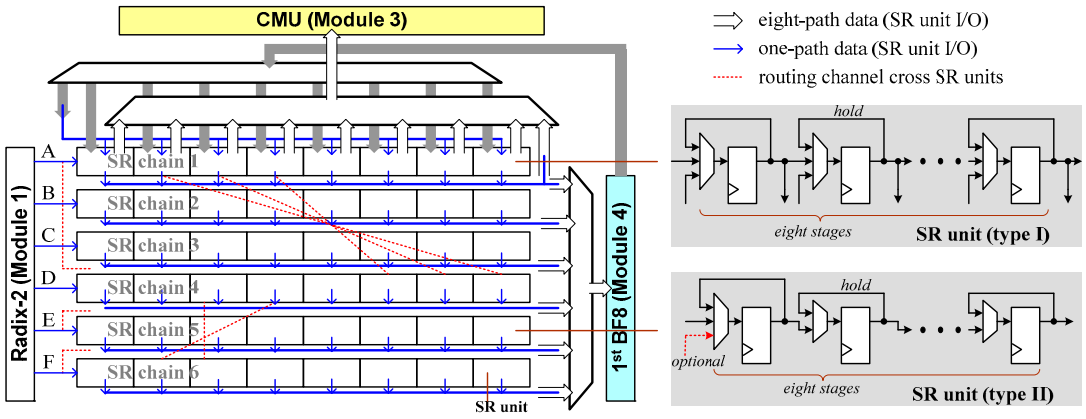


Figure 11. Block diagram of Module 2 (sorting buffer).

Figure 12 details the sorting operations of Module 2 corresponding to the periodic 10 TS units (i.e., TS0 to TS9) in Figure 8. In Figure 12, the data of each of the six A, B, . . . , or F streams from Module 1 (Figure 11) were represented using eight 8-sample data sections (i.e., X0, X1, . . . , and X7, where X can refer to A, B, . . . , or F). A, B, . . . , or F indicate the current six streams, whereas A′, B′, . . . , or F′ denotes the next six streams. Figure 12 at TS0 show that seven data sections of each stream (e.g., A0-A6 and B0-B6) were prepared in SR chains. From TS0 to TS5, the first BF8 operation was performed for the eight-section A, B, . . . , and F sequence, and the calculated results were stored back to SR chain 1. At next TSs (i.e., TS1–TS6), the first BF8 outcomes stored in SR chain 1 for B-F streams were sent out for CMU and for second BF8 operations. The data access schemes of SR chain 1 for first BF8 (stored in) and CMU/second BF8 (sent out) were alternately changed to satisfy the radix-8/8 data permutation [Equation (5)]. This observation was enabled using type I SR (Figure 11), which proved dual serial/parallel data shifts. Figure 12 shows the data of the new-coming six sequences, namely A′, B′, . . . , and F′, were sent to Module 2 after TS3. By appropriately routing the data across the SR chains (the red dashed line in Figures 11 and 12), the eight-section data of A′, B′, . . . , and F′ streams were regularly arranged in SR chains for new preparation at TS7/TS8/TS9. Therefore, the proposed sophisticated sorting buffer scheme allowed Module 2 to efficiently access intermediate data for radix-2/8/8 operations. The number of word storage elements for Module 2 was 64 per stream. By combing the level of two 32-element RAMs for Module 1, the total number of storage elements was 128. This process facilities the use of minimum memory sizes for 128-point FFT as the level of SDF/MDF-based designs [9,24–26].

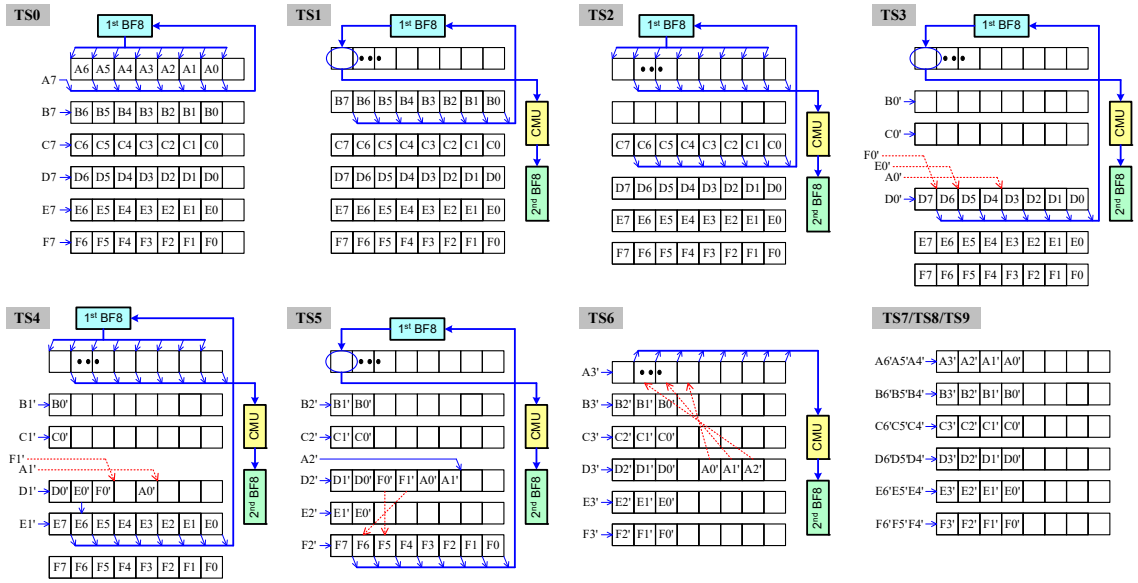


Figure 12. Details of operations in TS0-TS9 (Figure 8) for Module 2 (sorting buffer).

3.2.3. Module 3

Module 3 (CMU) performed intermediate W_{128} and W_{64} multiplications for the radix-2/8/8 algorithm. According to the $\pi/4$ symmetry of twiddle factors [32], W_{64} multiplications could be calculated using only eight factors of $W_{64}^{p'} = \exp(-j2\pi p'/64)$ and $p' = 1, 2, \ldots, 8$, in cooperation with sweeping and sign controls [24,25]. Considering the six eight-cycle TSs (i.e., TS1–TS6) executed by the second BF8 in Figure 12, the CMU/W_{64} multiplication for each cycle within a single TS period could be performed using the eight-path $W_{64}^{p'}$ factors, which is shown using p' parameters in Table 2. The appropriate rescheduling of cycles 2, 4, and 6 (the red arrow in Table 2) could be used to reduce conflict in $W_{64}^{p'}$ multiplication in order to minimize the number of required $W_{64}^{p'}$ constant multipliers. Additional multiplexer controls and registers were introduced to Module 2 to enable rescheduling. For W_{128} multiplication, $\pi/4$-symmetry check increased the required number of $W_{128}^{p'}$ factors to sixteen ($p' = 1, 2, \ldots, 16$) compared with $W_{64}^{p'}$. However, the scheme in [33] can be employed to decompose $W_{128}^{p'}$ into a form as Equations (6) and (7) if p' parameter is even or odd. Because $W_{64}^{k,k\pm1}$ could be obtained using one of the original $W_{64}^{p'}$ factors, thus, $W_{128}^{p'}$ could be derived as a combined calculation of W_{128}^{1or3} and $W_{64}^{p'}$. Therefore, only W_{128}^{1or3} constant multipliers were required to perform W_{128} multiplication with the existing $W_{64}^{p'}$ constant multipliers.

$$W_{128}^{p'} = W_{128}^{2k} = W_{64}^{k}, \textit{when } p' \textit{ is even} \tag{6}$$

$$\begin{aligned} W_{128}^{p'} = W_{128}^{2k+1} &= W_{128}^{1} W_{64}^{k} = W_{128}^{-1} W_{64}^{k+1} \\ &= W_{128}^{3} W_{64}^{k-1} = W_{128}^{-3} W_{64}^{k+1}, \textit{when } p' \textit{ is odd} \end{aligned} \tag{7}$$

As mentioned previously (Figures 9 and 10), six streams of data were accessed between Modules 1 and 3 for W_{128} multiplications. To avoid conflict operations associated with access to identical six-stream W_{128} factors in the same cycle, a one-sample shift was sequentially applied to the six paths of the CMU-related data accesses of the RAM banks (from Module 1, Figure 9). Figure 13 shows that the architecture of Module 3 allows access to six streams of data from Module 1 for multiplications involving W_{128} factors, and eight data paths from Module 2 for multiplications involving W_{64} factors (two arrow lines in Figure 13). Several duplicated W_{128}^{3} and W_{64}^{4} constant multipliers were used to manage residual conflicting W_{128} and W_{64} multiplications, such as cycle 4 in Table 2, to perform W_{64}^{4} multiplication twice. On the basis of our evaluation, the overall CMU (Module 3) covered an area

equivalent to 4.18 complex multipliers (structured as four real multipliers and two adders). This process was more area-efficient than the direct approach by using six/eight complex multipliers for six-stream or eight-path operations.

Table 2. Lists and scheduling of eight-path $W_{64}^{p'}$ factors (in p') for each cycle in a given TS.

→cycles	0	1	2	3	4	5	6	7
BF8 path 1	0	0	0	0	0	0	0	0
BF8 path 2	0	1	2	3	4	5	6	7
BF8 path 3	0	2	4	6	8	6	4	2
BF8 path 4	0	3	6	7	4	1	2	5
BF8 path 5	0	4	8	4	0	4	8	4
BF8 path 6	0	5	6	1	4	7	2	3
BF8 path 7	0	6	4	2	8	2	4	6
BF8 path 8	0	7	2	5	4	3	6	1

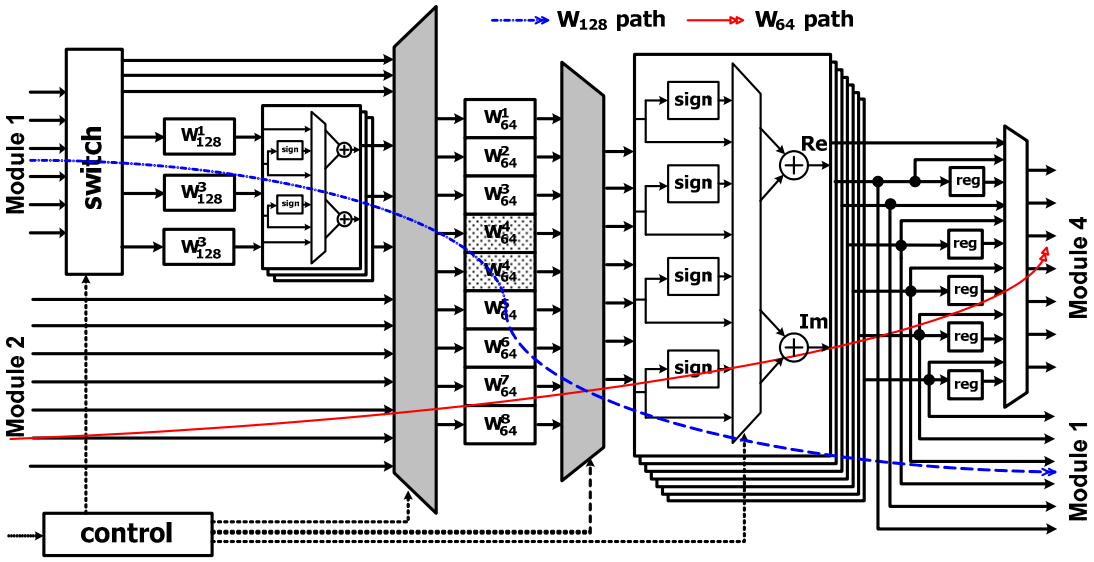


Figure 13. Block diagram of Module 3 (CMU).

3.2.4. Module 4

The two BF8 units in Module 4 were used for the first and second BF8 operations, respectively. Figure 14 shows the BF8 unit used three radix-2 stages based on the radix-2^3 algorithm [9]. In each radix-2 stage, four sets of BF2 elements were used for eight-path BF8 operations. Moreover, one W_8^1 and one W_8^3 constant multiplier each were required between the radix-2 stages 2 and 3.

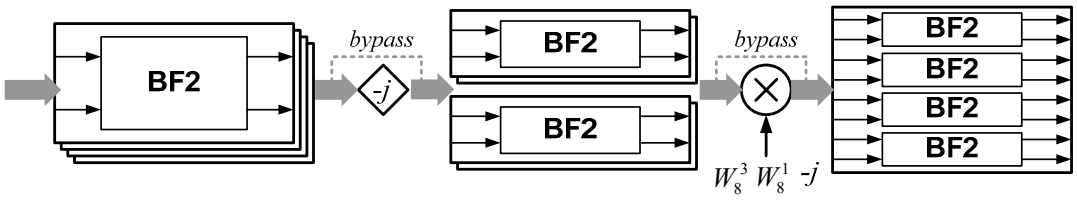
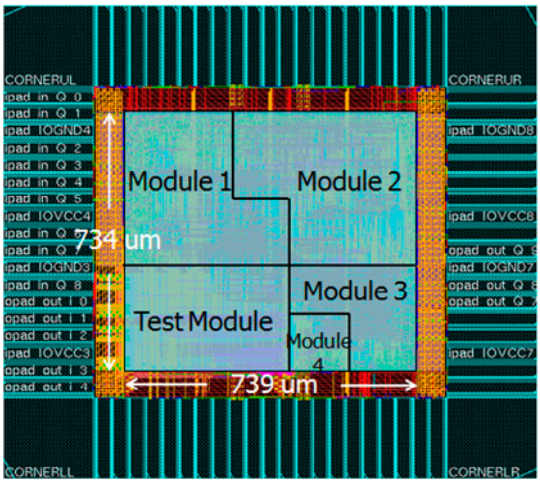


Figure 14. Block diagram of the first or second BF8 unit in Module 4.

4. Results and Comparison

4.1. Design Implementation

The evaluated signal is used to evaluate the proposed kernel accuracy. After fixed-point simulation using MATLAB, the signal-to-quantization-noise ratio (SQNR) was approximately 40 dB by using the proposed FFT kernel. This was implemented in a 1.0-V UMC 90-nm 1P9M complementary metal-oxide semiconductor process, the proposed FFT kernel used the Synopsys Design Compiler to synthesize the RTL code and employed Cadence SOC Encounter for placement and routing. The proposed FFT kernel consumed 10.72 mW of power and was operated at 80 MHz, and the core area of the proposed kernel was 739 × 734 μm^2. Figure 15 shows the core layout of the proposed kernel and its characteristics. The kernel test chip could perform 128/64-point FFT in 4–6 streams depending on the GI′ duration. A test module was included in this kernel chip to enable the serial storage of test patterns from the I/O and to send them to the kernel circuit along multiple paths. Moreover, the FFT results were sent to the test module and then sequentially read out through I/O for data evaluation. The post-layout simulation was performed based on the idea that the FFT kernel continuously processes the pattern stored in the test module.



FFT modes	6-stream / 128-point
Technology	UMC 90 nm
Core Voltage	1.0 Volt
Wordlength (IO/Internal)	12/16 bits
SQNR	40.17 dB
Clock Rate	80 MHz
Core Area	0.55 mm^2
Power Consumption	10.72 mW (@ 80 MHz)

Figure 15. Layout and performance summary of FFT kernel test chip.

4.2. Comparison

Table 3 compares the proposed design with previous studies which could perform the same FFT operations (i.e., maximum FFT length of 128). An SDF 128-point FFT processor was developed based on the radix-2^4 algorithm [30] and its performance was extended to four streams for evaluation. That which is compared deals with devices based on different technologies; therefore, the area was normalized [Equation (8)] based on the scheme in [27,33]. Although the GI duration is relatively short (e.g., 1/4 symbol duration) in timeliness considerations, the proposed FFT kernel could lend support to a throughput gain of (6/4) for the number of streams, only mainly needing additional memory elements for the two-stream data storage. The comparison results demonstrated the per-stream area efficiency of the proposed design by using area-efficient Modules 2 and 3 (sorting buffer and CMU), resource sharing and hardware use GI duration. Table 3 shows the improved area efficiency of the proposed design because the proposed scheme achieved the highest throughput by using modest hardware resources (i.e., throughput per area).

$$Nor.\,Core\,Area = \frac{Core\,Area}{(Tech./90\,nm)^2} \tag{8}$$

Table 3. Performance evaluation and comparison based on 128-point FFT specification.

	[25]	[30] (ref.)	[21]	Proposed
Architecture	MRMDF	R-2^4 SDF	MDC	MRM^{2}DF
Technology	0.13 µm	90 nm	0.18 µm	90 nm
No. stream (path)	4	4	4	6
Clock rate (MHz)	40	80	75	80
Nor. core area[3] (mm^2)	0.67 [1]	0.54	0.525 [2]	0.55 [1]
Throughput (*R*: clock rate)	4*R*	4*R*	4*R*	6*R* (six streams)
Throughput per area (*R*/mm^2)	5.97	7.41	7.62	10.91

[1] A test module was included. [2] An output sorting buffer was included. [3] All the values for area were normalized using Equation (8).

5. Conclusions

In this study, an area-efficient FFT kernel was presented using the MRM^{2}DF structure for multistandard MIMO-OFDM applications. The proposed design scheme allowed 64/128-point FFT in up to six streams through the efficient use of GI duration, thereby enhancing area efficiency per stream. Novel resource-sharing and operation-rescheduling schemes were developed using an area-reduced CMU to minimize multiplication hardware costs. Finally, a sophisticated sorting buffer was proposed using the minimum memory size to further reduce area overhead. A test chip was developed using UMC 90-nm technology, and was validated through post-layout simulation. The proposed design exhibited an area efficiency superior to that of previous 128-point-based FFT designs in terms of the throughput per area.

Author Contributions: S.-N.T. conceptualized and designed the proposed methodology. He also mainly wrote the manuscript. Y.-H.C. supported the technique consultation and co-wrote the article.

Funding: This work was supported in part by the Ministry of Science and Technology of Taiwan under project MOST 107-2221-E-033-009 and 107-2221-E-182-066, and the Chang Gung Memorial Hospital, Linkou under project CMRPD2H0301, CMRPD2H0051, CMRPD2G0312, and CIRPD2F0013.

Acknowledgments: The authors would like Hsiuan-Ting Wang and Chih-Chiao Tsai for materials preparation and technical supports in IC physical designs.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Yang, H.A. Road to future broadband wireless access: MIMO-OFDM-based air interface. *IEEE Commun. Mag.* **2005**, *43*, 53–60. [CrossRef]
2. Panajotović, A.; Riera-Palou, F.; Femenias, G. Adaptive uniform channel decomposition in MU-MIMO-OFDM: Application to IEEE 802.11ac. *IEEE Trans. Wirel. Commun.* **2015**, *14*, 2896–2910. [CrossRef]
3. Ketonen, J.; Juntti, M.; Cavallaro, J.R. Performance—complexity comparison of receivers for a LTE MIMO–OFDM system. *IEEE Trans. Signal Process.* **2010**, *58*, 3360–3372. [CrossRef]
4. Dahlman, E.; Parkvall, S.; Skold, J. *4G—LTE-Advanced Pro and the Road to 5G*; Academic Press: Cambridge, MA, USA, 2016.
5. Parkvall, S.; Dahlman, E.; Furuskar, A.; Frenne, M.N.R. The new 5G radio access technology. *IEEE Commun. Stand. Mag.* **2017**, *1*, 24–30. [CrossRef]
6. Ali, A.; Hamouda, W. A multi-mode IFFT/FFT processor for IEEE 802.11ac: Design and implementation. *Wirel. Commun. Mob. Comput.* **2016**, *16*, 1713–1725. [CrossRef]
7. Yang, C.H.; Yu, T.H.; Marković, D. Power and area minimization of reconfigurable FFT processors: A 3GPP-LTE example. *IEEE J. Solid-State Circuits* **2012**, *47*, 1–12. [CrossRef]
8. Baas, B.M. A low-power, high-performance, 1024-point FFT processor. *IEEE J. Solid-State Circuits* **1999**, *34*, 380–387. [CrossRef]

9. Shousheng, H.; Torkelson, M. Designing pipeline FFT processor for OFDM (de)modulation. In Proceedings of the URSI International Symposium on Signals, System, and Electron, Pisa, Italy, 22 October 1998; pp. 257–262.
10. Jo, B.G.; Sunwoo, M.H. New continuous-flow mixed radix (CFMR) FFT using novel in-place strategy. *IEEE Trans. Circuits Syst. Part I Regul. Pap.* **2005**, *52*, 911–919. [CrossRef]
11. Chen, C.M.; Hung, C.C.; Huang, Y.H. An energy-efficient partial FFT processor for the OFDMA communication system. *IEEE Trans. Circuits Syst. II Express Briefs* **2010**, *57*, 136–140. [CrossRef]
12. Mohanty, B.K.; Meher, P.K. Area-delay-energy efficient VLSI architecture for scalable in-place computation of FFT on real data. *IEEE Trans. Circuits Syst., Part-I Regul. Pap.* **2019**, *66*, 1042–1050. [CrossRef]
13. Tang, S.N.; Jan, F.C.; Cheng, H.W.; Lin, C.K.; Wu, G.Z. Multimode memory-based FFT processor for wireless display FD-OCT medical systems. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2014**, *61*, 3394–3406. [CrossRef]
14. Xing, Q.J.; Ma, Z.G.; Xu, Y.K. A novel conflict-free parallel memory access scheme for FFT processors. *IEEE Trans. Circuits Syst. II Express Briefs* **2017**, *64*, 1347–1351. [CrossRef]
15. Liu, S.; Liu, D. A high-flexible low-latency memory-based FFT processor for 4G, WLAN, and future 5G. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2018**, *27*, 511–523. [CrossRef]
16. Hwang, Y.T.; Chen, Y.J.; Chen, W.D. Scalable FFT Kernel designs for MIMO OFDM based communication systems. In Proceedings of the IEEE Conferrence TENCON, Taipei, Taiwan, 30 October–2 November 2007; pp. 1–4.
17. Hung, C.L.; Long, S.S.; Shiue, M.T. A low power and variable-length FFT processor design for flexible MIMO OFDM systems. In Proceedings of the IEEE International Symposium on Circuits and System (ISCAS), Taipei, Taiwan, 24–27 May 2009; pp. 705–708.
18. Chen, S.G.; Huang, S.J.; Garrido, M.; Jou, S.J. Continuous-flow parallel bit-reversal circuit for MDF and MDC FFT architectures. *IEEE Trans. Circuits Syst. I Reg. Pap.* **2014**, *61*, 2869–2877. [CrossRef]
19. Lin, Y.T.; Tsai, P.Y.; Chiueh, T.D. Low-power variable length fast Fourier transform processor. *Proc. IEE Comput. Dig. Tech.* **2005**, *152*, 499–506. [CrossRef]
20. Yang, K.J.; Tsai, S.H.; Chuang Gene, C.H. MDC FFT/IFFT processor with variable length for MIMO-OFDM systems. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2013**, *21*, 720–731. [CrossRef]
21. Fu, B.; Ampadu, P. An area efficient FFT/IFFT processor for MIMO OFDM WLAN 802.11n. *J. Signal Process. Syst.* **2009**, *56*, 59–68. [CrossRef]
22. Locharla, G.R.; Mahapatra, K.K.; Ari, S. Variable length mixed radix MDC FFT/IFFT processor for MIMO-OFDM application. *IET Comput. Digit. Tech.* **2018**, *12*, 9–19. [CrossRef]
23. Yoshizawa, S.; Orikasa, A.; Miyanaga, Y. An area and power efficient pipeline FFT processor for 8×8 MIMO-OFDM systems. In Proceedings of the IEEE International Symposium on Circuits and System (ISCAS), Rio de Janeiro, Brazil, 15–19 May 2011; pp. 2705–2708.
24. Lin, Y.W.; Liu, H.Y.; Lee, C.Y. A 1-GS/s FFT/IFFT processor for UWB applications. *IEEE J. Solid-State Circuits* **2005**, *40*, 1726–1735. [CrossRef]
25. Lin, Y.W.; Lee, C.Y. Design of an FFT/IFFT processor for MIMO OFDM systems. *IEEE Trans. Circuits Syst. I Reg. Pap.* **2007**, *54*, 807–815. [CrossRef]
26. Liu, H.; Lee, H. A high performance four-parallel 128/64-point radix-2^4 FFT/IFFT processor for MIMO-OFDM systems. In Proceedings of the IEEE Asia Pacific Conference on Circuits and System (APCCAS), Macao, China, 30 November–3 December 2008; pp. 834–837.
27. Chen, Y.; Lin, Y.W.; Taso, Y.C.; Lee, C.Y. A 2.4-Gsample/s DVFS FFT processor for MIMO OFDM communication systems. *IEEE J. Solid-State Circuits* **2008**, *43*, 1260–1273. [CrossRef]
28. Tang, S.N.; Liao, C.H.; Chang, T.Y. An area- and energy-efficient multimode FFT processor for WPAN/WLAN/WMAN systems. *IEEE J. Solid-State Circuits* **2012**, *47*, 1419–1437. [CrossRef]
29. Liu, L.; Ren, J.; Wang, X.; Ye, F. Design of low-power, 1GS/s throughput FFT processor for MIMO-OFDM UWB communication system. In Proceedings of the IEEE International Symposium Circuits and System (ISCAS), New Orleans, LA, USA, 27–30 May 2007; 2007; pp. 2594–2597.
30. Oh, J.Y.; Lim, M.S. Area and power efficient pipeline FFT algorithm. In Proceedings of the IEEE Workshop on Signal Processing Systems (SiPS), Athens, Greece, 2–4 November 2005; pp. 520–525.
31. Mahdavi, M.; Edfors, O.; Öwall, V.; Liu, L. A low latency FFT/IFFT architecture for massive MIMO systems utilizing OFDM guard bands. *IEEE Trans. Circuits Syst. I Reg. Pap.* **2019**, *66*, 2763–2774. [CrossRef]

32. Maharatna, K.; Grass, E.; Jagdhold, U. A 64-point Fourier transform hip for high-speed wireless LAN application using OFDM. *IEEE J. Solid-State Circuits* **2004**, *39*, 484–493. [CrossRef]
33. Tang, S.N.; Tsai, J.W.; Chang, T.Y. A 2.4 GS/s FFT processor for OFDM based WPAN applications. *IEEE Trans. Circuits Syst. II Express Briefs* **2010**, *57*, 451–455. [CrossRef]