Microprocessors and Microsystems 42 (2016) 227–234

Contents lists available at ScienceDirect

Microprocessors and Microsystems

journal homepage: www.elsevier.com/locate/micpro

A low-area dynamic reconfigurable MDC FFT processor design

Trong-Yen Lee*, Chi-Han Huang, Wei-Cheng Chen, Min-Jea Liu

Department of Electronic Engineering, National Taipei University of Technology, Taiwan

ARTICLE INFO

Article history:
Received 1 May 2015
Revised 29 January 2016
Accepted 1 February 2016
Available online 16 February 2016

Keywords:
OFDM
Fast Fourier Transform
Partial dynamic reconfiguration

ABSTRACT

Fast Fourier Transform (FFT) is widely utilized to perform data computation in orthogonal frequency-division multiplexing (OFDM) systems. Wireless networks use 64-point to 512-point FFT to implement task. However, different FFT protocols have different lengths. Building all required points into circuits independently leads to a massive hardware resource requirement. To minimize the hardware resource requirement, Partial dynamic reconfiguration (PDR) FPGA can be adopted to build FFT modules and switched using time-independent circuits, which increasing the system design flexibility. The proposed FFT processor uses 64-point FFT on a static circuit, with other points configured as reconfigurable modules (RMs). The system can configure 64 to 512 points under different environment, and using PDR also reduced the occupied hardware resources. This work presents a four-path Multipath Delay Commutator (MDC), which can increase the FFT processor throughput. The twiddle factor (T.F.) computational circuits in FFT can be implemented with a right shift and adder circuits to calculate the fixed-point format to minimize the design complexity. Experimental results illustrate that the proposed design can increase the throughput of FFT processor, and reduce the number of FPGA slices by up to 76.6% at 256-point FFT, and the flip-flop by up to 58.9% at 64-point FFT.

1. Introduction

Orthogonal frequency-division multiplexing (OFDM) is now widely used in communication protocols such as WPAN, WLAN and WMAN. The FFT is an important algorithm used in any OFDM implementation, but different communication systems usually have different FFT lengths and throughput requirements [1]. Because of the rapid development of communication technology, any one communication system must handle a wide variety of different standards. However, different communication protocols use FFTs of different sizes. If custom circuits for all different FFT sizes commonly used for OFDM algorithms are implemented on a chip, then the chip area would be very large.

FPGAs are now a common way of implementing circuitry. Because they are reconfigurable, FPGA devices provide designers with an environment for designing system prototypes, and enable manufacturers to release products without waiting for full ASIC respins. A partially reconfigurable FPGA device can reuse the same hardware resources in the reconfigurable area to perform multiple quite different tasks, switching on-the-fly from hardware optimized for one task to hardware optimized for another, while maintaining common elements statically across tasks. This partially reconfigurable technique (PRT) can improve hardware utilization while decreasing circuit area and system design cost.

This work uses PRT to implement FFT computational circuits, allowing us to create a variety of different FFT modules with the same FPGA hardware. We designed several different FFT RM in advance, and use a partially reconfigurable system to configure different reconfigurable FFT modules while the static circuit is still working, based on the requirements of particular FFTs. By allowing the same block of hardware to be customized appropriately, this PRT design methodology also reduces power consumption and hardware resources needed to achieve high performance.

To increase the execution speed, Multi-input Multi-output (MIMO) architectures are usually used to process FFT data. MIMO can promote data capacity and throughput under limited bandwidth, and MIMO-OFDM increases data transmission speed [2]. As a result, we designed a 4×4 MIMO Multipath Delay Commutator (MDC) FFT architecture for our architecture. For Discrete Fourier Transform (DFT), the Cooley and Tukey [3] proposed a method to reduce the computation complexity.

Implementing an FFT transformation requires multiplying twiddle factors in the FFT computational flow. However, a longer FFT requires more twiddle factors to be multiplied of varying size [4,5]. The Cho and Lee [6] proposed a modified Radix-25 FFT processor for high rate WPAN applications. We use the trigonometric

* Corresponding author. Tel.: +886 227712171.
E-mail addresses: tylee@ntut.edu.tw (T.-Y. Lee), kojoe23@hotmail.com (C.-H. Huang), t101418084@ntut.edu.tw (W.-C. Chen), david_liu@lct.com.tw (M.-J. Liu).

http://dx.doi.org/10.1016/j.micpro.2016.02.001
0141-9331/

complementary theorem to reduce the area required to implement the twiddle computational circuitry.

Finally, this work applies two advanced circuits in the FFT computation to reduce the area for large systems. The first circuit replaces multiplying circuits with right-shift and adder logic for twiddle factors, to reduce the area. In the second circuit, the architecture stores the multi-point Radix N FFT circuits in memory, which can run-time context-switch other FFT circuits through DR to reduce the FPGA area.

The remaining of this paper is organized as follows. Section 2 describes related work. Section 3 introduces the proposed new low-area dynamic reconfigurable MDC FFT processor. Experimental results are shown in the Section 4. Section 5 draws conclusions.

2. Related work

The FFT computation depends on different radix algorithms and architectures. In order to meet the communication requirement, different architectures have been developed to implement corresponding communication condition. Discrete Fourier Transform (DFT) is usually adopted to process digital signals, but it has high computational complexity, and takes a long time to implement. Therefore, Cooley and Tukey proposed in 1965 [3] a method to reduce the computational complexity of DFT. FFT computation rules follow Radix-n, where a higher n indicates more complex computational rules and circuits. A Radix-n algorithm can compute an FFT that has a point size of a power of any multiple of n. Therefore, if the required computation is 512-point FFT, the function needs to be implemented with both Radix-2 and Radix-4. The Radix-2 algorithm reduces DFT computational complexity from $O(N^2)$ to $O(N\log N)$ [4,5].

Many FFT architectures have been designed, including memory-based architecture, cached-memory architecture and pipelined architecture [7]. The pipelined architecture has a high throughput, and its hardware area and memory can be reduced by using different FFT algorithms. Therefore, pipeline is suitable for OFDM. The pipeline architecture can be classified as Multipath Delay Commutator (MDC) and Single-path Delay Feedback (SDF) [8]. SDF has single input and single output, and so requires FFT computation to be performed in a specific order. Therefore, the system needs to store the earlier input data in a register element; wait for the calculation of the input data to complete, and then transmit the output to the next stage. A basic N-point SDF has several computational stages in N-point FFT computation, and each stage contains a register element, a butterfly (BF) circuit and a twiddle-factor computational circuit. The register element of each stage decreases at each computational stage. After BF computation, the output data are multiplied by the corresponding twiddle factors, then transmitted to the next stage. The butterfly circuits and register elements are each changed by a different radix. The complex multiplier number of N-point SDF is $(\log_2 N)-1$. The butterfly circuits have a complexity of $\log_2 N$, and require $(N-1)$ register elements. However, the throughput and speed of FFT computation are insufficient to meet the requirement. Therefore, the MDC architecture has been proposed to obtain high efficiency. The SDF architecture is less efficient than the MDC architecture. The MDC architecture is multi-path, and therefore can input multiple FFT data simultaneously. The complexity of N-point MDC multiplier is $(\log_2 N)-1$; that of the butterfly circuits is $\log_2 N$, and the number of register elements is $(3N/2)-1$. The comparison of property with SDF and MDC are shown in Table 1. The implementation tools for FFT with FPGA device are shown in [9,10,11].

Vennila et al. [12] presented a scalable and runtime dynamically reconfigurable FFT architecture for different wireless standards. The method applies modified SDF to design 64, 128 and 256-point FFT. The hardware resource with multiplexer based reconfigurable FFT and SDF architecture are not enough to meet current communication requirements. Wang and Liu [13] and Liu and Lee [14] utilized ROMs to store twiddle factors, and used 4 multipliers to handle data. Rudagi et al. [15] used real part and imaginary part circuits to implement the twiddle factors. All the comparisons of different FFT-point experimental result are shown in Section 4. The 64-point hardware resources are shown Table 5. Chitra and Srivatsa [16] developed a MIMO-OFDM mix-radix 4-path MDC architecture using 128- and 64-point FFT implemented on Xilinx 3s400pq208-5, and the 128-point hardware resources are shown Table 5. Dreschmann et al. [17] presented a 256-point FPGA-based FFT algorithm, Chao et al. [18] presented a reconfigurable FFT design, namely Spatio-Temporally-shared Reconfigurable Fast Fourier Transform, for the digital audio broadcasting (DAB) protocol in radio stations. The 256-point hardware resources are shown Table 5. Karachalios et al. [19] developed four SDF architectures using Radix-2 and implemented on Xilinx Virtex-4 SX35, and the 512-point hardware resources are shown Table 5.

The hardware circuits of Radix-2 are simple to design. However, Radix-2 needs many computational stages to implement N-point FFT. This work presents the Radix-4 module, which reduces the number of computational stages. The Radix-4 module works with four data at the same time, and has a higher throughput than Radix-2 [6]. Radix-4 has more complicated control circuits than Radix-2: it calculates four input data at the same time, and separates N-point DFT into 4 $N/4$-point portions.

Table 1
Comparison of property with SDF and MDC.

	SDF	MDC
Number of complex multipliers	$(\log_2 N)-1$	$(\log_2 N)-1$
Number of butterfly circuits	$\log_2 N$	$\log_2 N$
Number of register elements	$(N-1)$	$(3N/2)-1$

3. Design of low-area 4-path dynamic reconfigurable MDC FFT processor

This section describes the design and implementation of a 4-path dynamic reconfigurable (FPDR) MDC FFT processor. The goal of this design is to increase throughput and reduce the size of the FFT processor by adopting a dynamic reconfigurable device. The twiddle factor generation has been redesigned to simplify the representation and implementation of the circuits. The design adopts dynamic reconfigurable technique to minimize hardware resources.

3.1. FFT computational circuit

Eq. (1) shows a basic N-point DFT computation format, where $x(n)$ denotes the input sequence in the time domain; $X(k)$ denotes the output sequence in the frequency domain, and N represents the transformation length. The symbol W_N represents the twiddle factor, which is the twiddle denotes the Nth primitive root of unity. The FFT computation follows the radix-r algorithm, where r denotes the computation amount r of FFT data. We chose to use a Radix-4 FFT equation and circuit, much like that described in [2]. Each 64-point stage requires $N_s = N/r^s$, where $0 \le s \le 2$, which computation flow is shown in Fig. 1, and the required twiddle factors also different. Radix-4 divides Eq. (1). The Radix-4 64-Point FFT twiddle factors are shown in Eq. (2). If N is not a power of r, then a mixed-radix algorithm is required to meet the requirement. Eq. (3) shows the output data with real and imaginary parts.

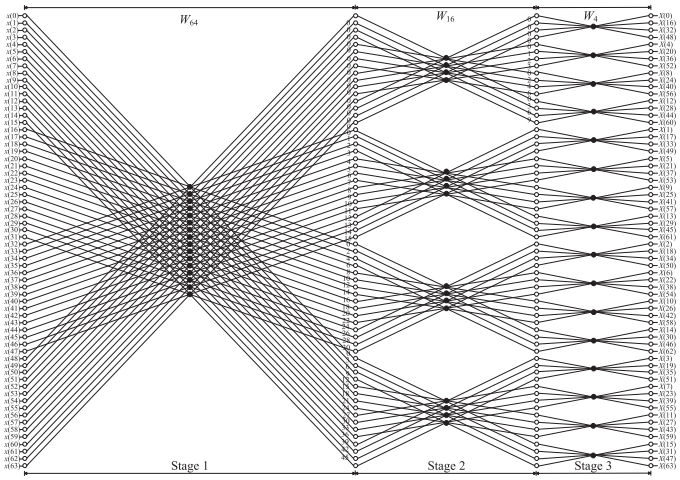

Fig. 1. Radix-4 64-point butterfly.

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk}, \text{ where, } W_N^{nk} = e^{-j(2\pi nk/N)},$$
$$k = 0, 1, \ldots, N-1$$
$$n = 16n_1 + 4n_2 + n_3, \text{ where, } 0 \le n_1, n_2, n_3 \ge 3$$
$$k = k_1 + 4k_2 + 16k_3, \text{ where, } 0 \le k_1, k_2, k_3 \ge 3 \quad (1)$$

$$W_{16}^{nk} = e^{-j(2\pi nk/16)} = \cos(2\pi nk/16) - j\sin(2\pi nk/16) \quad (2)$$

$$\begin{aligned} \text{OutputData} &= \text{InputData} \times W_{16}^{nk} \\ &= (\text{Re}_{in} + \text{Im}_{in}) \times (\cos(2\pi nk/16) - j\sin(2\pi nk/16)) \\ &= (\text{Re}_{in}\cos(\pi nk/8) + \text{Im}_{in}\sin(\pi nk/8)) \\ &\quad + j(\text{Im}_{in}\cos(\pi nk/8) - \text{Re}_{in}\sin(\pi nk/8)) \end{aligned} \quad (3)$$

An FFT computational circuit has three parts, namely the commutator, data computational and twiddle factor computational circuits. The proposed FFT computational circuit utilizes a 4×4 MIMO processor, which is shown in Fig. 2. It can be expanded to the required point size using the same design method.

Fig. 3 illustrates the point-changing condition of 64-point FFT in the commutator circuit. FFT and the three different FFT points are computed in a specific order. The first input data need to be commuted, and then output with appropriate data to the next computational stage. Because the input data need to be commuted, each computational stage has a delay in each path in Radix-4 FFT. Fig. 4 shows the 4-Path Radix-4 FFT delay path, which is $N/4$, $N/2$ and $3N/4$ respectively of the data of each path, where N denotes N-point FFT. Fig. 5 illustrates a commutation rule in a Radix-4 FFT circuit. The input data are stored in the respective delay register, and the output then connects to the commutator to place data in the proper computation order. Fig. 6 shows the four steps of the commutator, namely input, input delay, commutator and output delay, and the changes to the 64-point input data at each step. In the input stage, the processor waits for the 4-path data to be processed which store in the register. In the register diagram, each if the four colors represented different paths. When the 4-path data are ready to be processed, a delay equivalent to four paths is applied to the pre-pipeline in the input delay stage, as shown in the left half of Fig. 4. The pipeline in the commutator stage is divided into seven sections. Fig. 5 shows the commutation rule in a Radix-4 FFT circuit. When the 4-path data commutation complete, a of the four-paths-circuit delay is applied to integrate the commutation data in the output delay stage, as shown in the right half of Fig. 4.

A Radix-4 computation flow is proposed in Fig. 7. Four data of four paths are computed, namely (1,1,1,1), (1,−j,−1,j), (1,−1,1,−1)

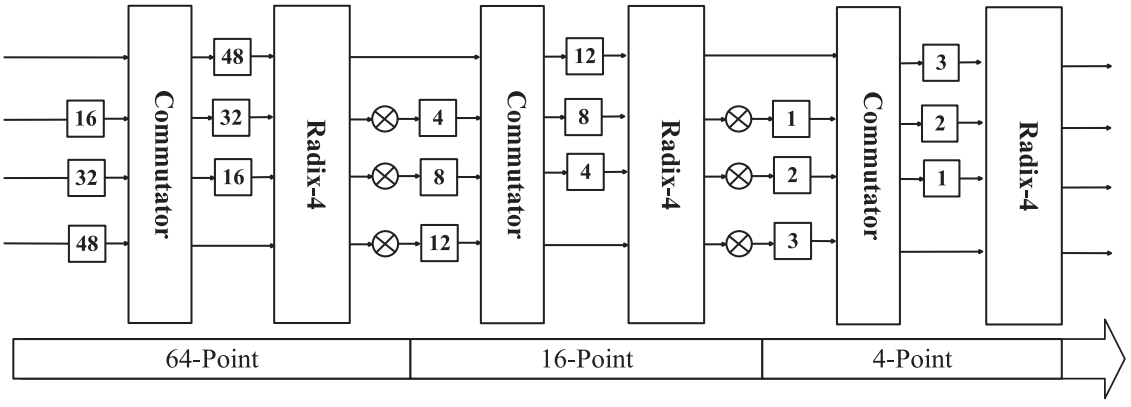

Fig. 3. The changing condition of 64-point FFT.

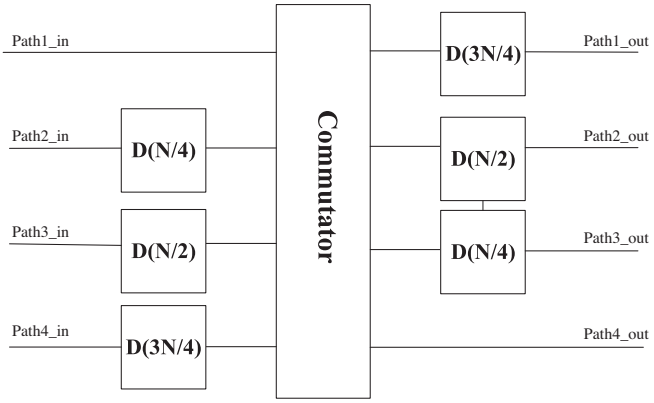

Fig. 4. Delay of four paths.

Fig. 2. 64-point architecture.

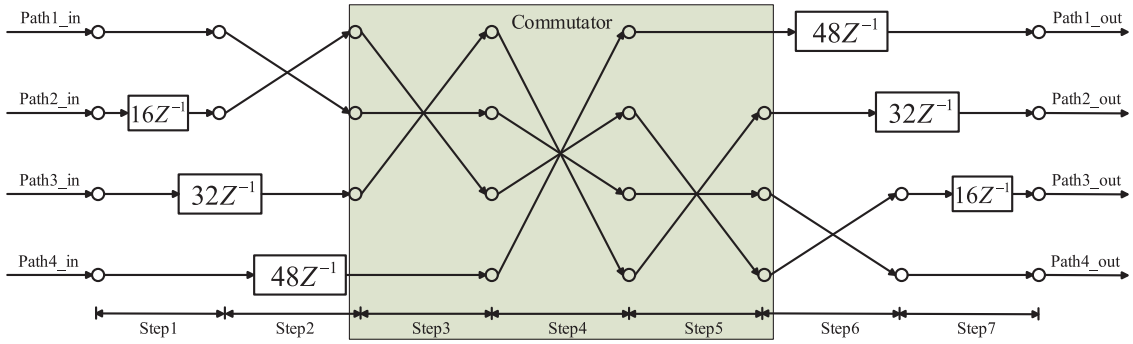

Fig. 5. Commutation rule of Radix-4.

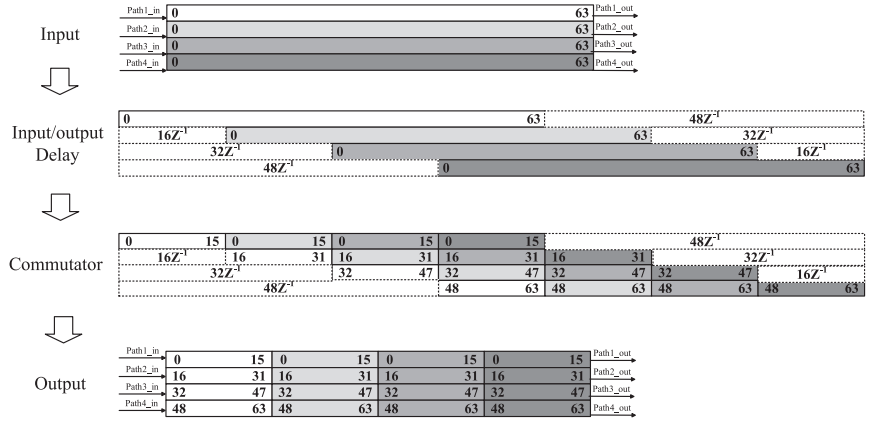

Fig. 6. The changing condition of input data.

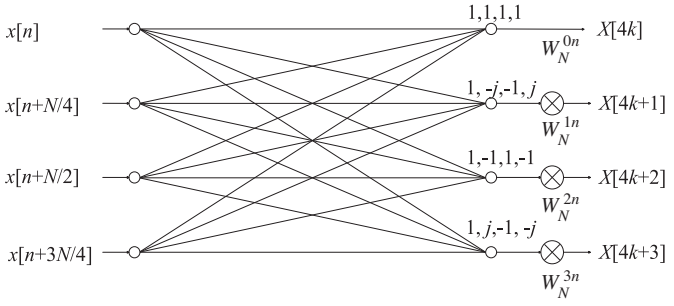

Fig. 7. Radix-4 computational flow.

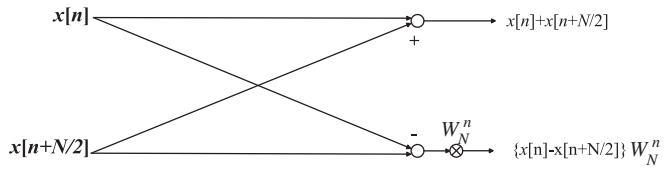

Fig. 8. Radix-2 computational flow.

and (1,*j*,−1,−*j*). Twiddle computational circuits multiply the outputs. The FFT-point need to compute Radix-2 which computational flow is shown in Fig. 8. Two-input data be computed then the results will be outputted to next computational stage.

The twiddle computation in each FFT-point is performed in the twiddle factor computational circuits. The proposed FFT architecture reduces the hardware resources of twiddle circuits because by reducing the area required by the real part of Twiddle and imaginary part of Twiddle selectors. The twiddle factor of the 16-point FFT can be separated into two parts, cosθ and sinθ, as described in Eq. (2) The output data is derived as in Eq. (3). The two parts of cosθ and sinθ complement each other, as shown in Table 2. To show the value of cosθ and sinθ, Eq. (4) gives an example showing cos$2\pi/8$ derived as an exponent. The multiplier operation is then performed using the right-shift circuits to complete computation. Fig. 9 shows the right shift circuit.

Table 2
The values of cosθ and sinθ in 16-point FFT twiddle factors.

T.F.	*nk*	*a*	cos(*a*)	sin(*a*)
$W^0{}_{16}$	0	0	1	0
$W^1{}_{16}$	1	$\pi/8$	0.9238	0.3826
$W^2{}_{16}$	2	$2\pi/8$	0.7071	0.7071
$W^3{}_{16}$	3	$3\pi/8$	0.3826	0.9238
$W^4{}_{16}$	4	$4\pi/8$	0	1
$W^6{}_{16}$	6	$6\pi/8$	−0.7071	0.7071
$W^9{}_{16}$	9	$9\pi/8$	−0.9238	−0.3826

The values of sinθ and cosθ require computation in twiddle factors on 16-point FFT. Fig. 10 illustrates the proposed simplified twiddle factor computational circuits that implement these equations. The computational cycle controller in the circuit computes the twiddle factors in corresponding to each cycle. The proposed method employs a simple circuit comprising right-shift and adder logic to calculate cosθ of floating-point values. This right-shift and adder configuration can replace the traditional multiplying circuit to reduce the area before the Re-Twiddle and Im-Twiddle selectors are applied.

$$\cos 2\pi/8 = \sqrt{2}/2 = 0.7071 = 2^{-1} + 2^{-3} + 2^{-4} + 2^{-6} + 2^{-8} + 2^{-14} \quad (4)$$

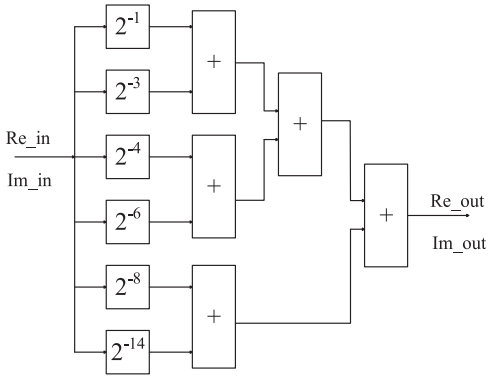

Fig. 9. cos2π/8 right shift circuit.

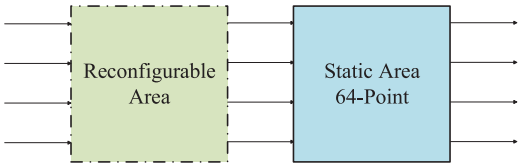

Fig. 11. Reconfigurable circuit design.

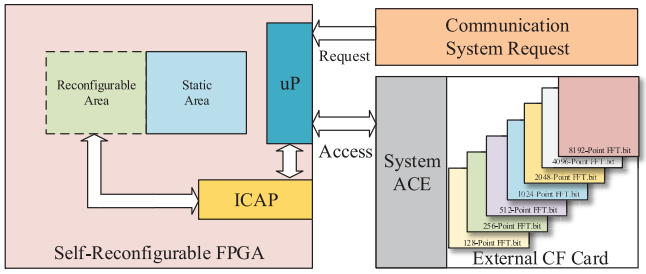

Fig. 12. 4-path dynamic reconfigurable MDC FFT processor architecture.

3.2. RMs design

A communication system has many protocols, and handles signals using orthogonal frequency-division multiplexing (OFDM). In OFDM block, a significant block FFT is in charge of transferring the modulated signal from the time domain to the frequency domain. The shortest FFT length for OFDM is 64 points. Therefore, this work adopts a 64-point static circuit in a reconfigurable system, which can be expanded to longer FFT by configuring RMs into the reconfigurable region.

As we described earlier in Section 3.1, *N*-point FFTs are decomposed into several stages using power-of-2 Radix factors that are divisors of *N*. The computation of FFT normally follows the radix algorithm. Radix-X can compute the number of FFT point by power of X. In general, mixed-radix computation is more flexible than single-radix. Thus, 128-point FFT could be computed using Radix-2 and Radix-4 together. We design and implement 64-point to 512-point FFT circuits with an FPGA reconfigurable function. The proposed FFT architecture contains a reconfigurable area and a static area, as shown in Fig. 11. The 64-Point FFT circuit is used as the static circuit in static area, and others circuit functions are implemented in the reconfigurable area where used for dynamic reconfiguration of the required FFT computation corresponds to the reconfigurable radix modules.

The reconfigurable radix module is arranged into overall circuits of different FFT points, which are shown in Table 3. Realizing a reconfigurable functionality requires building a reconfigurable system on FPGA. The proposed method uses module concept to design cores for different FFT-point RMs. Fig. 12 shows the proposed architecture of the system. The RMs are stored in the external compact flash card, and the microprocessor (μP) is a Xilinx-supported MicroBlaze which is adopted to control the System Advanced Configuration Environment (System ACE), and communicates with the ICAP Processor through the ICAP Interface. The system also has a MicroBlaze Debug Module (MDM), which enables JTAG-based debugging of one or more MicroBlaze processors, thus helping to

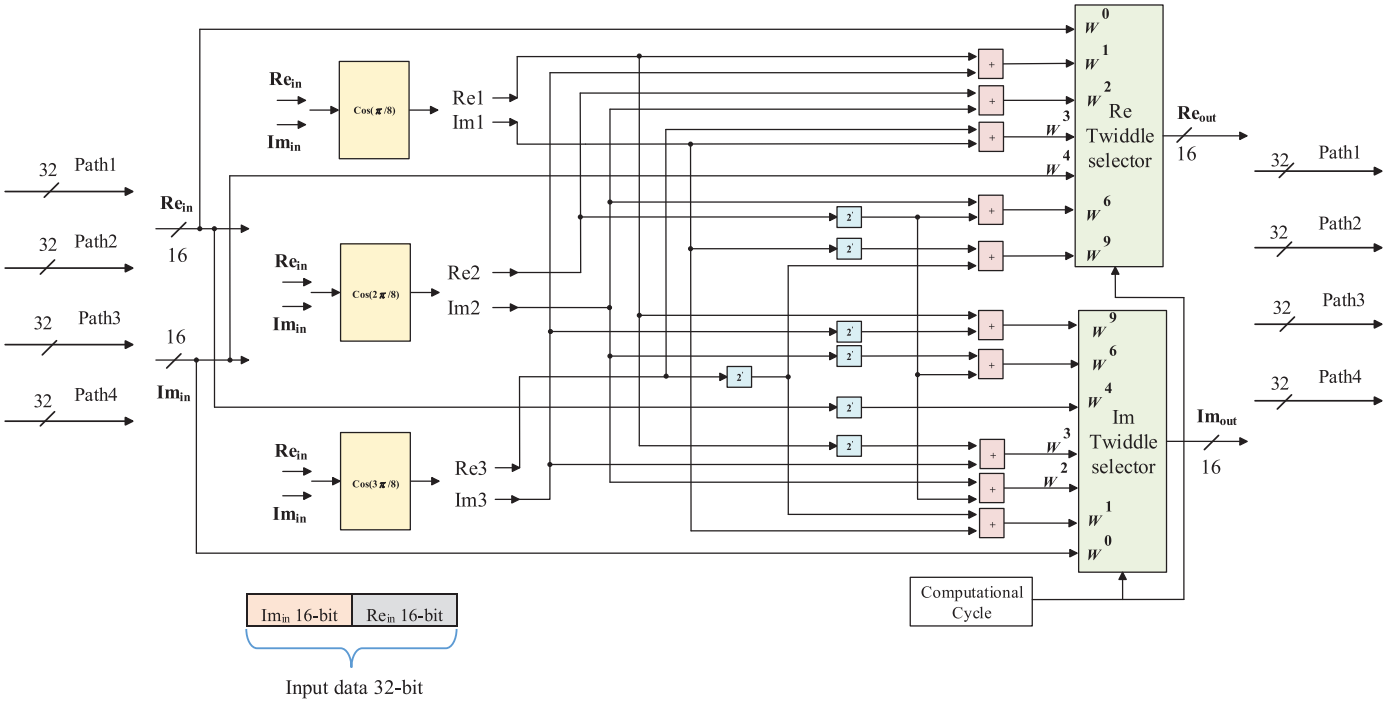

Fig. 10. Twiddle factor computational circuit.

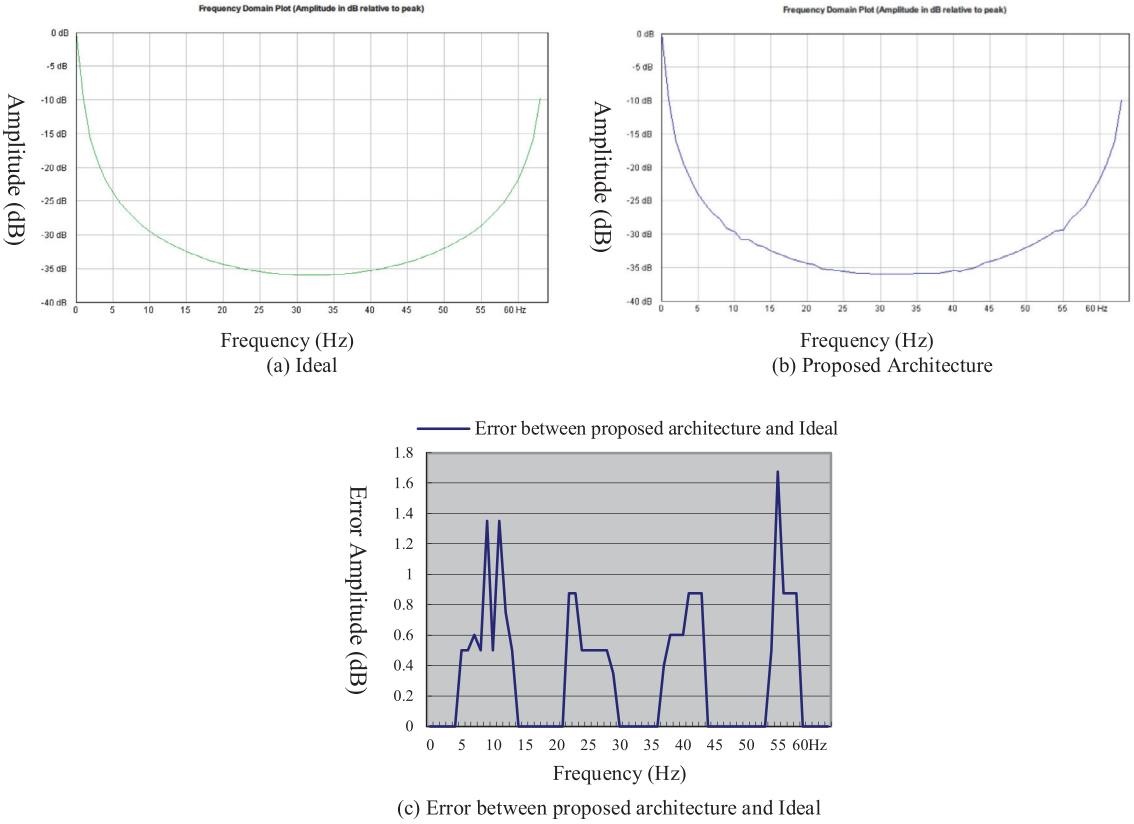

Fig. 13. Amplitude vs. frequency on FFT computation.

Table 3
Radix arrangement of FFT computational circuit.

FFT-point	Reconfigurable circuits				Static circuits		
64-point					Radix-4	Radix-4	Radix-4
128-point				Radix-2	Radix-4	Radix-4	Radix-4
256-point				Radix-4	Radix-4	Radix-4	Radix-4
512-point			Radix-4	Radix-2	Radix-4	Radix-4	Radix-4
1024-point			Radix-4	Radix-4	Radix-4	Radix-4	Radix-4
2048-point		Radix-4	Radix-4	Radix-2	Radix-4	Radix-4	Radix-4
4096-point		Radix-4	Radix-4	Radix-4	Radix-4	Radix-4	Radix-4
8192-point	Radix-4	Radix-4	Radix-4	Radix-2	Radix-4	Radix-4	Radix-4

simplify the design. The top module is separated into static and RMs. The static module is set as 64-point FFT, and the RMs are configured from 128-point to 512-point FFT.

To implement the system, bitstream files of RMs need to be stored in an external CF card. This work uses 128-point to 512-point RMs. To improve flexibility, multiple bitstream files can be stored in a portable storage device such as a compact flash memory card. The combination of the System ACE CF controller and a standard CompactFlash device delivers a powerful configurable solution for high-density FPGA systems. Finally, the FFT can be applied in a large system such as OFDM [20], which includes Serial to Parallel Conversion, QAM Modulation and FFT module. The OFDM in the digital systems connects to other DSP process units.

4. Experimental results

We designed FPDR by using ISE Design Suite 13.2 of Xilinx. The platform of implementation FPGA used Xilinx Virtex-5, and the chip was CVLX110T-FF1136, then we examined the function correction of the proposed architecture through an online tool [11], and the simulation results are shown in Fig. 13(a). The curve produced by the proposed architecture is shown in Fig. 13(b), and is similar to the ideal curve. The simulation results of each path in FFT were stored in files, and compared with ROMs data. The different values of error between proposed architecture and Ideal shown in Fig. 13(c).

Table 4
Comparison of throughput.

	Proposed	Wang [13]
Throughput	400 M/s	200 M/s
Critical path	8.959 ns	–
Reduced (%)		50

In the testing stage, the real and imaginary values of twiddle factor were stored in two testing ROMs. The data of simulation results, a reconfigurable system file, a static area and a reconfigurable area were stored in ROMs. The floorplan condition was stored in the PlanAhead tool. The experimental processing is described as follows. First, the PlanAhead tool generated the partial reconfigurable bit files and system image. The RMs files were then loaded into the CF card. Finally, the FPGA chip of the Virtex-5 was

Table 5
Comparison of hardware resource.

64-point FFT				
	Proposed	Wang [13]	Liu [14]	Patil [15]
Architecture	MDC	MDC	MDC	MDC
Word length	16	8	6	16
Parallel former	4 × 4	8 × 8	4 × 4	–
Radix	4	2	2^4	2
Complex multiplier	0	4	4	0
Number of slice	2652	3596	3470	–
Reduced (%)	–	26.3	23.6	–
Number of flip flops	2349	5717	–	5600
Reduced (%)	–	58.9	–	58.1
128-point FFT				
	Proposed		Chitra [16]	
Architecture	MDC		MDC	
Word length	16		8	
Number of slice	2965		3303	
Reduced (%)	–		10.2	
Number of flip flops	3358		5649	
Reduced (%)	–		40.6	
256-point FFT				
	Proposed (FPDR)	Proposed (without reconfigurability)	Meyer [17]	Chao [18]
Word length	16	–	6	16
Radix	4	–	4/2	2
Number of FPGA slices	11,295	23,961	48,335	37,680
Reduced (%) (FPDR)	–	52.9	76.6	70
Reduced (%) (FPDR without reconfigurability)	–	–	50.4	36.4
Reduced by reconfigurability architecture (%)	–	–	26.2	33.6
512-point FFT				
	Proposed (FPDR)	Proposed (without reconfigurability)	Karachalios [19]	
Radix	4	–	2	
Parallel former	4 × 4	–	4 × 4	
Number of FPGA slices	20,670	33,498	21,624	
Reduced (%) (FPDR)	–	38.3	4.4	
Reduced (%) (FPDR without reconfigurability)	–	–	−35.44	
Reduced by reconfigurability architecture (%)	–	–	4.4	

employed to implement and emulate this experiment. The Xilinx ChipScope Analyzer tool was used to observe the waveform results.

Table 4 shows the comparisons of throughput with 64 to 256-point FFT [13], indicating that by using right shift and adder circuits to replace multiplication circuits, which can improve calculation time and critical path, this method improves throughput by up to 50%. The critical path, as measured by the ISE timing report, is 8.959 ns, representing a maximum frequency of about 111.6 MHz.

In ordinary design upper point than 64-point FFT circuits that need to contain the 64-point FFT in every circuit, which means all the required hardware resources plus more times of 64-point FFT, and result in hardware consumption. But, in proposed reconfigurable system, 64-point is considered as a static circuit which is a fixed circuit, and only one time of resource. All we need to do is to configure the corresponding point RMs to satisfy the computation. That is why we consider the 64-point FFT not a part of RMs which are independent circuits.

Table 5 contains all the comparison results from 64-512bit between proposed architecture and related works. The proposed FPDR 64-point FFT stores the repeat area into ROM and uses content-switching method to change the hardware at run time. Comparison of hardware resource used with related works [13,14,15] are shown that the proposed method can reduce the FPGA slices by up to 26.3% and flip-flops by up to 58.9%. The comparisons of 128-point FFT with related work [16] indicate that the proposed FFT method can reduce the FPGA slices by up to 10.2% and flip-flops by up to 40.6%.

The FPDR reduces the area by reconfiguring the commutator and twiddle factor circuit. The 64-point FFT is based on static circuits without unused reconfigurability architecture. The FPDR and FPDR without reconfigurability architecture (i.e. only commutator and twiddle factor circuit) were compared to measure the saved area that the proposed design provides. The proposed FFT method compares with 256-point FFT [17,18], indicating that the FPDR reduces FPGA slices by up to 76.6%, of which 26.2% is contributed by the reconfigurability architecture. The proposed FFT method compares with 512-point FFT [19], indicating that the reconfigurability architecture in the proposed FFT method reduces FPGA slices by up to 4.4%.

Table 6
FFT configuration time.

	128-point	256-point	512-point
Configuration time (ms)	0.569	0.623	0.685

The proposed method reduces the area used by twiddle factors, and reduces the FPGA area in FFT computation using a dynamic reconfigurable method. The overhead of the proposed method is that reconfiguration for FFT need a few time to configure the FPGA device. Table 6 shows the penalty times, indicating that the configuration time increases in direct proportion with the FFT size. A typical application system needs to be run one or two times to reconfigure it from one orientation to another. Therefore, the required reconfiguration time is just under 5% of execution time.

5. Conclusion

This work presents a low-area dynamic reconfigurable MDC FFT processor for wireless networks. Computation is performed using a shifter instead of complex multipliers. The real and imaginary parts of the twiddle factors are turned into a set, which is re-

peatedly reused, reducing the design complexity of the proposed system. The proposed system also implements FFT architecture with the partial reconfigurability, which reduces hardware circuits, power consumption, hardware resource and occupied area. The designed circuits are classified as static or reconfigurable. If time-independent circuits need to be added in to the architecture, then they can be simply created as RMs. Using partial reconfigurable technique, the proposed method can switch the FFT-Point computation between different RMs, thus increasing the flexibility of the system. The proposed method increases throughput and reduces the area used by the FFT processor using a dynamic reconfigurable device. The twiddle factors circuit used right shift can simplify the representation, and reduces the used area. This FFT design uses a PDR-FPGA device to minimize the hardware resources. Although the proposed method has a time penalty, it is more effective than previous methods. Future work will be to design and implement a large-scale FFT for high-bandwidth communication. Finally, experimental results illustrate that the proposed design can increase the throughput of FFT processor, and reduce the number of FPGA slices by up to 76.6% at 256-point FFT, and the flip-flop utilization by up to 58.9% at 64-point FFT.

Acknowledgements

The authors would like to thank the reviewers who given very valuable comments for improving the quality of this paper. The authors want to thank the Ministry of Science and Technology of the Republic of China, Taiwan, partially supported this work under Contract no. MOST 103-2221-E-027-125.

References

[1] S.N. Tang, C.H. Liao, T.Y. Chang, An area- and energy-efficient multimode FFT processor for WPAN/WLAN/WMAN systems, IEEE J. Solid State Circuits 47 (6) (2012) 1419–1435.
[2] Kang, B. Kim, J., Low complexity multi-point 4-channel FFT processor for IEEE 802.11n MIMO-OFDM WLAN system, in: Proceedings of the IEEE International Symposium on Green and Ubiquitous Technology (GUT), Jul. 2012, pp. 94–97.
[3] J.W. Cooley, J.W. Tukey, An algorithm for the machine calculation of complex Fourier series, in: Proc. Math. Comput., 19, 1965, pp. 297–301.
[4] E.J. Kim, H.S. Myung, High speed eight-parallel mixed-radix FFT processor for OFDM systems, in: Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS), 2011, pp. 1684–1687.
[5] S. Yoshizawa, A. Orikasa, Y. Miyanaga, An area and power efficient pipeline FFT processor for 8×8 MIMO-OFDM systems, in: Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS), 2011, pp. 2705–2708.
[6] T. Cho, H. Lee, A high-speed low-complexity modified Radix-25 FFT processor for high rate WPAN applications, IEEE Trans. Very Large Scale Integr. (VLSI) Syst. 21 (1) (2013) 187–191.
[7] C.C. Hung, P.L. Chiu, Y.H. Huang, A variable-length FFT processor for 4×4 MIMO-OFDM systems, in: Proceedings of the IEEE International Symposium on VLSI Design Automation and Test (VLSI-DAT), 2010, pp. 156–159.
[8] B.C. Lin, Y.H. Wang, J.D. Huang, J.Y. Jou, Expandable MDC-based FFT architecture and its generator for high-performance applications, in: Proceedings of the IEEE International SOC Conference (SOCC), 2010, pp. 188–192.
[9] http://www.xilinx.com, Partial Reconfiguration User Guide, UG702 (v14.5), 2013 (accessed 15.07.14).
[10] http://www.xilinx.com, Partial Reconfiguration of Xilinx FPGAs Using ISE Design Suite, WP374 (v1.2), 2012 (accessed 18.08.14).
[11] http://www.random-science-tools.com/index.html, Fast Fourier Transform Calculator, 2015 (accessed 25.03.15).
[12] C. Vennila, G. Lakshminarayanan, Seok-Bum Ko, Dynamic partial reconfigurable FFT for OFDM based communication systems, Circuits Syst. Signal Process. (CSSP) 31 (3) (2012) 1049–1066.
[13] X. Wang, Y. Liu, Special-purpose computer for 64-point FFT based on FPGA, in: Proceedings of the IEEE International Symposium on Wireless Communications and Signal Processing (WCSP), 2009, pp. 1–3.
[14] H. Liu, H. Lee, High-speed four-parallel 64-Point radix-2^4 MDF FFT/IFFT processor for MIMO-OFDM systems, in: Proceedings of the International Technical Conference on Circuits/Systems, Computers and Communications, July 6–9, Shimonoseki, Japan, 2008, pp. 1469–1472.
[15] J.M. Rudagi, R. Lobo, P. Patil, N. Biraj, N. Nesaragi, An efficient 64-point pipelined FFT engine, in: Proceedings of the IEEE International Symposium on Advances in Recent Technologies in Communication and Computing (ARTCom), 2010, pp. 204–208.
[16] M.P. Chitra, S.K. Srivatsa, Design of low power mixed radix FFT processor for MIMO OFDM systems, in: Proceedings of the IEEE International Symposium Advances in Computing, Control, & Telecommunication Technologies (ACT '09), 2009, pp. 2705–2708.
[17] M. Dreschmann, J. Meyer, M. Hubner, R. Schmogrow, D. Illerkuss, J. Ecker, J. Euthold, W. Freude, Implementation of an ultra-high speed 256-point FFT for Xilinx Virtex-6 devices, in: Proceedings of the IEEE International Symposium on Industrial Informatics (INDIN), 26-29, 2011, pp. 829–834.
[18] H.L. Chao, C.Y. Peng, C.C. Wu, K.S. Huang, C.H. Lu, J.S. Shen, P.A. Hsiung, Spatio-temporally-shared reconfigurable fast Fourier transform architecture design, in: Proceedings of the IEEE International Symposium on Field-Programmable Technology, (FPT.2013), 9-11 Dec, 2013, pp. 426–429.
[19] A. Karachalios, K. Nakos, D. Reisis, H. Alnuweiri, A new FFT architecture for 4×4 MIMO-OFDMA systems with variable symbol lengths, in: Proceedings of the IEEE International Symposium on Innovations in Information Technology (IIT.2009), 15-17 Dec, 2009, pp. 80–84.
[20] A. Mecwan, D. Shah, Implementation of OFDM Transceiver on FPGA, Proc. Nirma Univ. Int. Conf. Eng. (NUiCONE) 28–30 (2013) 1–5.

Trong-Yen Lee received his Ph.D. degree in Electrical Engineering from National Taiwan University, Taipei, Taiwan, ROC, in 2001. Since 2002, he has been a member of the faculty in the Department of Electronic Engineering, National Taipei University of Technology, where he is currently a professor. His research interests include hardware-software co-design of embedded systems, FPGA systems design, and VLSI design.

Chi-Han Huang received the M.S. degree in Electronic Engineering from National Taipei University of Technology, Taipei, Taiwan, ROC, in 2011. He is currently working toward the Ph.D. degree in the Department of Electronic Engineering, National Taipei University of Technology, Taipei, Taiwan. His current research interests include VLSI design and Network-on-Chip design.

Wei-Cheng Chen received the M.S. degree in the Graduate Institute of Computer and Communication from National Taipei University of Technology, Taipei, Taiwan, ROC, in 2014. His current research interests include VLSI design and reconfigurable FFT processor design.

Min-Jea Liu He is currently working toward the Ph.D. degree in the Department of Electronic Engineering, National Taipei University of Technology, Taipei, Taiwan, ROC. His current research interests include VLSI design and multiplier design.