A Combined SDC-SDF Architecture for Normal I/O Pipelined Radix-2 FFT

Zeke Wang, Xue Liu, Bingsheng He, Feng Yu

***Abstract*—We present an efficient combined SDC-SDF (Single-Path Delay Commutator-Feedback) radix-2 pipelined Fast Fourier Transform (FFT) architecture which includes log$_2$N-1 SDC stages, and 1 SDF stage. The SDC processing engine (PE) is proposed to achieve 100% hardware resource utilization by sharing the common arithmetic resource in the time-multiplexed approach, including both adders and multipliers. Thus, the required number of complex multipliers is reduced to log$_4$N-0.5, compared with log$_2$N-1 for the other radix-2 SDC/SDF architectures. Moreover, the proposed architecture requires roughly minimum number of complex adders log$_2$N+1 and complex delay memory 2N+1.5log$_2$N-1.5.**

***Index Terms*—FFT, Pipelined Architecture, Single-path Delay Communicator Processing Engine (SDC PE)**

I. Introduction

Fast Fourier Transform (FFT) has played a significant role in digital signal processing field, especially in the advanced communication systems such as orthogonal frequency division multiplexing (OFDM) [1] and asymmetric digital subscriber line (ADSL) [2]. All these systems require that the FFT computation must be high-throughput and low-latency. Therefore, designing a high-performance FFT circuit is an efficient solution to the above mentioned problems. Especially, the pipelined FFT architectures have mainly been adopted to address the difficulties due to their attractive properties, such as small chip area, high throughput, and low power consumption.

To the best to our knowledge, two types of pipelined FFT architectures can be found in the literature: delay feedback (DF) and delay commutator (DC). Further, according to the number of input data stream paths, they can be classified into multiple-path (M) or single-path (S) architectures. The two classifications form four kinds of pipelined FFT architectures (e.g., SDC). Multiple-path (M) architectures [3-9], are often adopted when the throughput requirement is beyond the theoretical limitation that the single-path architecture can offer at a given clock frequency. However, they require concurrent read (write) operations for the multi-path input (output) data. Therefore, single-path (S) architectures could be appropriate in some cases when the system cannot ensure concurrent operations. However, the arithmetic utilization is relatively low, compared with 100% utilization of the existing MDF/MDC architectures (for example, [4]). In the paper, we focus on the SDC radix-2 pipelined FFT architecture which can also achieve 100% multiplier utilization by re-ordering the inner data sequence.

For single-input data stream, the conventional radix-2 SDF FFT architecture [10] requires 2log$_2$N complex adders and log$_2$N-1 complex multipliers, where N is the FFT size. Both Chang et al. [11] and Liu et al. [12] present the novel SDC architectures to reduce 50% complex adders by re-ordering inner data sequences. However, the utilization of the corresponding complex multipliers still remains 50% for the both architectures. We therefore study whether the complex multiplier unit can be modified to achieve the 100% utilization.

In the radix-2 FFT architectures, there is a common observation that one half data (sum part of butterfly operation) do not involve complex multiplication (W_N^0) at all, while the other half (difference part) indeed involves complex multiplication (W_N^k). Hence, it has the opportunity to achieve the objective that reduces the arithmetic resource of the conventional complex multipliers by a factor of 2, leading to 100% utilization. It is ideal for two consecutive complex input data to contain a complex number which needs to execute complex multiplication. If so, we can minimize the re-ordering memory requirement while achieving the above objective that reduces 50% the arithmetic resource of complex multipliers.

Fortunately, the improved SDC architecture can produce the sum and the corresponding difference results of a butterfly operation in consecutive two cycles. The sum part is directly passed to the next stage, while the difference part needs to execute complex multiplication before passing to the next stage. Therefore, the SDC architecture is ideal for our efficient pipelined radix-2 FFT architecture. However, the SDF architecture does not meet the above constraint well since the sums of the all butterflies in the stage are produced first, followed by the corresponding differences.

In the paper, we present an efficient combined SDC-SDF radix-2 pipelined FFT architecture which includes log$_2$N-1 *SDC stages*, 1 *SDF stage*, and 1 *bit reverser*. The single-path delay commutator processing engine (*SDC PE)* in each *SDC stage* achieves the 100% hardware resource utilizations of both adders and multipliers. We include the *SDF stage* to re-order the data sequence, and then the delay memory of the *bit reverser* is reduced to N/2. The proposed architecture can

Manuscript received. This work was supported by the National Science and technology support program of China (No. 2012BAK24B01)

Zeke Wang is with Zhejiang University and Nanyang Technological University. (e-mail: wangzeke@zju.edu.cn)

Xue Liu is with Institute of Cyber-Physical Systems Engineering, College of Information Science and Engineering, Northeastern University Shenyang, Liaoning, China (e-mail: liuxue0412@tom.com).

Bingsheng He is with the School of Computer Engineering, Nanyang Technological University, Singapore, 639798. (e-mail: BSHE@ntu.edu.sg).

Feng Yu is with the Institute of Digital Technology and Instrument, Zhejiang University, Hangzhou, 310027, China. (e-mail: osfengyu@zju.edu.cn)

produce the same normal output order as [26].

II. Review of Pipelined FFT Architecture

A. FFT Review

The N-point DFT is defined by

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

where x(n) is the input data, W_N^{nk} is the coefficient($W_N^{nk} = e^{-2\pi nk/N}$) and N is any integer power of two.

It is well known that the radix-2 FFT can be deduced from DFT by factorizing the N-point DFT recursively into many 2-point DFTs. The data flow graph (DFG) of 16-point radix-2 FFT is shown in Fig. 1.

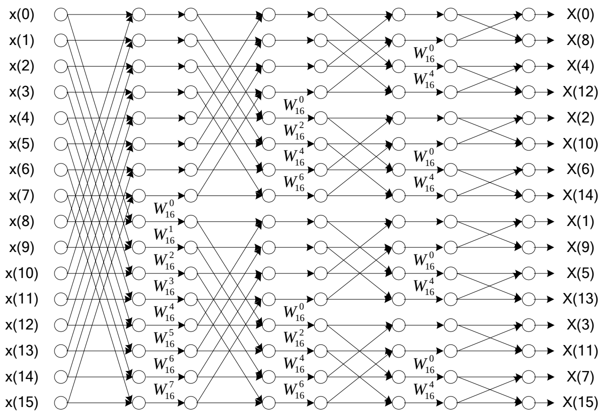

Fig.1 Data flow graph of DIF Radix-2 FFT (N = 16)

B. Review of Pipelined FFT Implementations

Assuming that the input data enters the FFT circuit serially in a continuous flow, the radix-2 MDC and SDF architectures can be directly deduced according to the data flow graph in Fig.1. The radix-2 MDC architecture [9] is the most direct implementation approach of pipelined FFT, but its hardware utilization is only 50%. Compared with [9], the radix-2 SDF design [10] reduces the required memory size. However, the utilizations of adders and multipliers are still 50%.

Besides the basic radix-2 architectures, various high radix pipelined FFT architectures have also been proposed to address the arithmetic resource utilization problem. They are radix-4 MDC [4, 8, 23], radix-4 SDC [13], radix-4 SDF [18], radix-2^2 SDF[14,21], radix-2^3 SDF [15], radix-2^4 SDF [16], radix-2^5 SDF [17], radix-r^k SDC/SDF [19] and radix-2^k feedforward [20]. Compared with the radix-2 architectures, the high radix architecture can only process the FFT, whose size is a power of its high radix, not just 2.

In order to extend the application scope of the FFT architectures, the new dynamic data scaling architectures [22] for pipelined FFTs have been proposed to implement both 1-D and 2-D applications. The MDC-based FFT architecture [23] has been proposed for the MIMO-OFDM systems with variable length. Employing folding transformation and register minimization techniques, the novel parallel pipelined architecture [24] for complex and real valued FFT has been proposed to significantly reduce power consumption.

III. A Combined SDC-SDF Radix-2 Pipelined FFT

For single-input data stream, we propose an efficient combined SDC-SDF radix-2 pipelined FFT architecture, and the proposed *SDC PE* structure can reduce 50% complex multipliers.

A. The Proposed FFT Architecture

The proposed FFT architecture consists of 1 *pre-stage*, $\log_2$N-1 *SDC stages*, 1 *post-stage*, 1 *SDF stage*, and 1 *bit reverser*, shown in Fig. 2(a). The *pre-stage* shuffles the complex input data to a new sequence that consists of real part followed by the corresponding imaginary part, shown in Table Ⅰ. The corresponding *post-stage* shuffles back the new sequence to the complex format. The *SDC stage t* (t=$1,2,\cdots,\log_2 N-1$) contains a *SDC PE*, which can achieve 100% arithmetic resource utilization of both complex adders and complex multipliers. The last stage, *SDF stage*, is identical to the radix-2 SDF, containing a complex adder and a complex subtracter. By using the modified addressing method [12], the data with an even index are written into memory in normal order, and they are then retrieved from memory in bit-reversed order while the ones with an odd index are written in bit-reversed order. Final, the even data are retrieved in normal order. Thus, the *bit reverser* requires only N/2 data buffer.

Table Ⅰ illustrates the inner data sequence of 16-point FFT computation. The complex input data at cycle *m* are (*m_r*, *m_i*), where *m_r* and *m_i* (m = 0, 1… 15) represent the real and imaginary parts, respectively. We only include the *pre-stage*, *SDC stage 1*, *2*, *3*, and *post-stage*, since the *SDF stage* has the same sequence as the *post-stage* except the 8-cycle delay, and the *bit reverser*, 8-cycle delay over the *SDF stage* [12], produces normal output sequence.

B. The Single-path Delay Commutator Processing Engine

The *SDC PE*, shown in Fig. 2(b), consists of a *data commutator*, a *real add/sub unit*, and an *optimum complex multiplier unit*. In order to minimize the arithmetic resource of the *SDC PE*, the most significant factor is to maximize the arithmetic resource utilization via re-ordering the data sequences of the above three units.

In the *stage t*, the *data commutator* shuffles its input data (*Node_A*) to generate a new data sequence (*Node_B*), whose index difference is N/2^t, where *t* is the index of stage. The new data sequence (*Node_B*) is critical to the *real add/sub unit*, where one real adder and one real subtracter can both operate on two elements for each input data. The sum and difference results (*Node_C*) overlap the places of the two input elements. Therefore, it preserves the data sequence, requires only one real adder and one real subtracter.

For the *optimum complex multiplier unit*, its output data sequence (*Node_E*) should be the same as its input data sequence (*Node_C*). If so, its output sequence (*Node_E*), which is also the output sequence of the *SDC stage t*, can become the direct input data sequence (*Node_A*) of the *SDC stage t+1*. The implementation detail is descripted in the next subsection.

Table Ⅱ illustrates the data sequence of *SDC stage 1* of 16-point FFT computation, including the data sequences of the

Table Ⅰ DATA OUTPUT ORDER OF THE PROPOSED PIPELINED ARCHITECUTRE FOR 16-POINT FFT

Cycle	pre-stage	stage 1	stage 2	stage 3	post-stage
1	0_r, 1_r	-	-	-	-
2	0_i, 1_i	-	-	-	-
3	2_r, 3_r	-	-	-	-
…	…	…	…	…	…
9	8_r, 9_r	-	-	-	-
10	8_i, 9_i	0_r, 8_r	-	-	-
11	10_r, 11_r	0_i, 8_i	-	-	-
12	10_i, 11_i	2_r, 10_r	-	-	-
13	12_r, 13_r	2_i, 10_i	-	-	-
14	12_i, 13_i	4_r, 12_r	-	-	-
15	14_r, 15_r	4_i, 12_i	0_r, 4_r	-	-
16	14_i, 15_i	6_r, 14_r	0_i, 4_i	-	-
17	-	6_i, 14_i	2_r, 6_r	-	-
18	-	1_r, 9_r	2_i, 6_i	0_r, 2_r	-
19	-	1_i, 9_i	8_r, 12_r	0_i, 2_i	0_r,0_i
20	-	3_r, 11_r	8_i, 12_i	4_r, 6_r	2_r,2_i
21	-	3_i, 11_i -	10_r, 14_r	4_i, 6_i	4_r,4_i
…	…	…	…	…	…
32	-	-	-	13_r, 15_r	11_r, 11_i
33	-	-	-	13_i, 15_i	13_r, 13_i
34	-	-	-	-	15_r, 15_i

The *pre-stage* shuffles the complex input data to a new sequence that consists of real part followed by the corresponding imaginary part, and *post-stage* undoes the data shuffling carried out by the *pre-stage*. The *SDC stage t (t=1, 2, 3),* preserves the data sequence generated by its *data commutator*.

above three units.

C. The Optimum Complex Multiplier Unit

As shown in Fig. 2(b), it contains 2 multiplexers (*M1* and *M2*), 1.5-word memory (*G1*, *G2* and *G3*), 2 *Real Multipliers* and 1 *Real Adder*. The signal *s* controls the behavior of two multiplexers (*M1* and *M2*): through or swap. The signal *s* also controls the behavior of the *Real Adder*, which supports both addition and subtraction operations.

For the input couple (*0_r,8_r*) and (*0_i,8_i*) at the *Node_C* in Table Ⅱ, the sum part data *0_r* and *0_i* will directly pass to the delay memory *G1* to generate *0_r** and *0_i** with one cycle delay in consecutive two cycles, while the difference part *8_r* and *8_i* will directly enter the *Real Multipliers* (*Node_D*) to generate (*c×8_r*, *d×8_r*) and (*c×8_i*, *d×8_i*) before reordering. The reordering process is performed as follows.

1) In the first cycle, when *8_r* comes, the signal *s* (*s* = 1) selects "through"; that is, the up (down) input of the multiplexer (*M1* or *M2*) connects to the up (down) output. Then, the *G2* (or *G3*) would be *d×8_r* (or *c×8_r*) in the second cycle.
2) In the second cycle, when 8_i comes, the signal *s* (*s* = 0) selects "swap"; that is, the up (down) input of the multiplexer (*M1* or *M2*) connects to the down (up) output. Then, the *G2* (or *G3*) would be *c×8_i* (or *d×8_r*) in the third cycle. The *s* will make the *Real Adder* perform subtraction operation and then c×8_r-d×8_i (8_r*) would appear at the *Node_E*.
3) In the third cycle, the signal *s* (*s* = 1) selects "through" for *M1* and *M2,* and chooses addition operation for *Real Adder*. Then, *d×8_r+c×8_i* (*8_i**) would appear at the *Node_E*.

Consequently, the complex result data couple (*0_r**, *8_r**) and (*0_i**, *8_i**) would come out at *New_Label* (*Node_E*) with one clock delay in consecutive two cycles.

The above mechanism can be iterated by applying to the other couples in the *stage 1*, e.g., (2_r, 10_r) and (2_i, 10_i), and so on. If we carry the above process towards the log_2N-1 *stages* to completion, we can complete the majority part of the radix-2 FFT computation.

In summary, the *SDC PE* can reduce 50% the arithmetic resource of complex multipliers in the time-multiplexing approach, at the expense of 1.5 complex delay memory overhead for each *SDC PE*.

IV. Comparison and Analysis

Table Ⅲ presents the hardware requirement of our design and the other pipelined architectures. The '*internal memory*' denotes the complex internal memory and the '*overall memory*' shows the complex total memory when the *bit reverser* is included. The typical SDF design requires the minimum overall memory 2N. The '*overall memory*' of the proposed design is $2N+1.5\log_2N-1.5$. It includes: (a), N-2 for the data commutators in the *SDC stages;* (b), $1.5\log_2N-1.5$ for the *optimum complex multiplier unit*s to re-order inner data sequences; (c), 2 for the pre-stage and post-stage; (d), N/2 for the *SDF stage*; and (e), N/2 for the *bit reverser*.

Table Ⅲ also lists the required numbers of complex adders and complex multipliers. The proposed design requires roughly minimum number $\log_2N+1$ of complex adders, and requires only $\log_4N-0.5$ complex multipliers compared with $\log_2N-1$ for the other radix-2 designs. The multiplier requirement is approximately as same as radix-2^2 [14, 21], and more than $R2^3$SDF [14] and $R2^4$SDF [16], since the high radix designs theoretically require fewer multipliers than the radix-2 designs. The proposed design preserves the radix-2 nature and achieves 100% multiplier utilization, while the other radix-2 designs only achieve 50% utilization $((2^1-1)/2^1)$. Furthermore, the high radix designs require more complex adders than the proposed design (except $R2SD^2F$ [21]), and can only process the FFT, whose size is a power of its high radix. For example, the 128-point FFT cannot be directly mapped to either one high radix design [14, 15, 16], but the radix-2 design can. Beyond the scope of the paper, the mixed radix design can implement the 128-point FFT with relatively complex control logic.

Since all of the FFT designs are single-path, their throughput is 1/N. Since the latency is roughly proportional to the size of the *overall memory*, the latency of the proposed design is $2N+\log_2N-1$. The critical path delay of the proposed design is $T_M + 2T_A + 3T_{MUX}$, where T_M, T_A and T_{MUX} are computation time of a multiplier, adder and multiplexer, respectively. Since the T_{MUX} is greatly less than T_M and T_A, the critical path delay of all designs is roughly same.

In the following, we consider a 256-point pipelined FFT with word length 16 bits. The multiplier, adder, and 16-bit SRAM are taken to be 4153, 505, and 96 transistors [25], respectively.

We compare the proposed design, in terms of transistor, with the other FFT designs, shown in Table Ⅳ. We observe that the proposed design requires fewer transistors than the other radix-2 architectures [11, 12] because of the reduction in complex multipliers. It is roughly the same as that of radix-2^2,

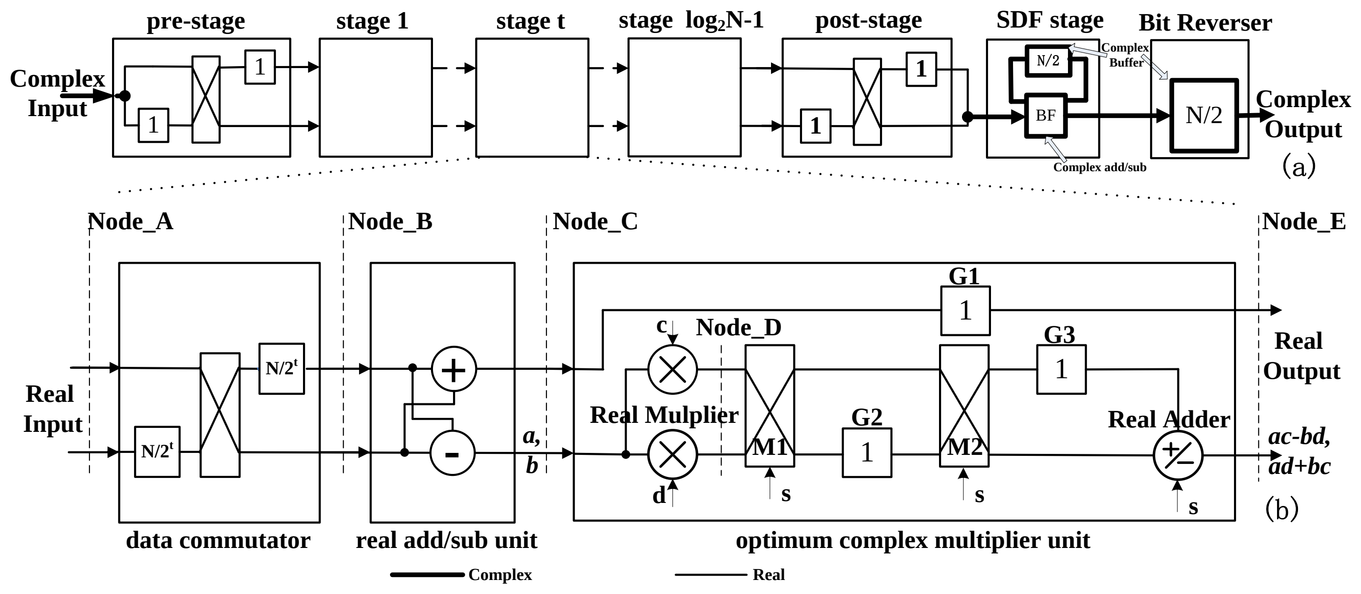

Fig.2 (a) Block diagram of the proposed FFT architecture, (b) Block diagram of the *SDC PE*, including a *data commutator*, a *real add/sub unit* and an *optimum complex multiplier unit*. '*a*' ('*b*') means the real (imaginary) part of subtraction result, '*c*'('*d*') means the real (imaginary) part of the twiddle factor. '*G1*', '*G2*' and '*G3*' mean three one-cycle delay elements. The signal's' controls the behavior of two multiplexers (*M1* and *M2*) and the *Real Adder*. When'*s*' is 1 (0), both multiplexers perform "through" ("swap") and the *Real Adder* performs addition (subtraction) operation.

TABLE II DATA SEQUENCE OF THE PROPOSED STAGE 1 OF 16-POINT FFT

Cycle	Node_A	Node_B/C	Node_D	G1	G2	G3	s	Real Adder	Node_E	New_Label
1	0_r, 1_r	-	-	-	-	-	-	-	-	-
2	0_i, 1_i	-	-	-	-	-	-	-	-	-
…	…	…	…	…	…	…	…	…	…	…
9	8_r, 9_r	0_r, 8_r	c×8_r ,d×8_r	-	-	-	-	-	-	-
10	8_i, 9_i	0_i, 8_i	c×8_i ,d×8_i	0_r	d×8_r	c×8_r	0	sub	0_r ,c×8_r- d×8_i	0_r* , 8_r*
11	10_r, 11_r	2_r, 10_r	c×10_r ,d×10_r	0_i	c×8_i	d×8_r	1	add	0_i ,c×8_i+d×8_r	0_i* , 8_i*
12	10_i, 11_i	2_i, 10_i	c×10_i ,d×10_i	2_r	d×10_r	c×10_r	0	sub	2_r ,c×10_r- d×10_i	2_r* , 10_r*
13	12_r, 13_r	4_r, 12_r	c×12_r ,d×12_r	2_i	c×10_i	d×10_r	1	add	2_i ,c×10_i+d×10_r	2_i* , 10_i*
14	12_i, 13_i	4_i, 12_i	c×12_i ,d×12_i	4_r	d×12_r	c×12_r	0	sub	4_r ,c×12_r- d×12_i	4_r* , 12_r*
15	14_r, 15_r	6_r, 14_r	c×14_r ,d×14_r	4_i	c×12_i	d×12_r	1	add	4_i ,c×12_i+d×12_r	4_i* , 12_i*
16	14_i, 15_i	6_i, 14_i	c×14_i ,d×14_i	6_r	d×14_r	c×14_r	0	sub	6_r ,c×14_r- d×14_i	6_r* , 14_r*
…	…	…	…	…	…	…	…	…	…	…
24	-	7_i, 15_i	c×15_i ,d×15_i	7_r	d×15_r	c×15_r	0	sub	7_r ,c×15_r- d×15_i	7_r* , 15_r*
25	-	-	-	7_i	c×15_i	d×15_r	1	add	7_i ,c×15_i+d×15_r	7_i* , 15_i*

For the *stage 1*, *Node_A* denotes the sequence just after the *pre-stage*. *Node_B* denotes the new sequence after the *data commutator* shuffles the data, whose index difference is N/2. *Node_C* denotes the data sequence after the *real add/sub unit*. *Node_D* denotes the difference part of *real add/sub unit*, multiplied by the twiddle factor (*c*+*d*i), where '*c*' ('*d*') stands for the real (imaginary) part of twiddle factor for each data. '*G1*'is used to delay the sum part of *real add/sub unit*. '*G2*' and '*G3*' have been used to re-order the internal data sequence. The signal's' controls the behaviors of the *Real Adder*, "add" or "sub", and the two multiplexers (*M1* and *M2*), "through" or "swap". *Node_E*, equal to "*New_Label*" (m_r* and m_i*), denotes complex butterfly computation result. "*New_Label*" is just the *Node_A* for the next stage (e.g., *stage 2*).

Table III HARDWARE RESOURCE COMPARISON FOR THE VARIOUS PIPELINED FFT ARCHITECTURES

Architecture	Internal Memory	Overall Memory	Adder	General Multiplier (Utilization)	Constant Multiplier	Throughput	Latency	Critical Path Dealy
R2SDF[10]	N-1	2N-1	$2log_2N$	$2log_4N$-1 (50%)	-	1/N	2N-1	$T_M + 2T_A + T_{MUX}$
R2SDC[19]	2N-2	3N-2	$2log_2N$	$2log_4N$-1 (50%)	-	1/N	N	$T_M + 2T_A + 2T_{MUX}$
Chang[11]	1.5N	2N	log_2N+1	$2log_4N$-1 (50%)	-	1/N	2N	$T_M + 2T_A + T_{MUX}$
Liu[12]	1.5N+2×(log$_2$N-2)	2N+2×(log_2N-2)	log_2N+1	$2log_4N$-1 (50%)	-	1/N	2N+2×(log_2N-2)	$T_M + 2T_A + 2T_{MUX}$
R2^2SDF [14]	N-1	2N-1	$2log_2N$	log_4N-1 (75%)	-	1/N	2N-1	$T_M + 2T_A + T_{MUX}$
Yang [21]	4(N-1)/3	(7N-4)/3	log_2N	log_4N-1 (75%)	-	1/N	(7N-4)/3	$T_M + 2T_A + 4T_{MUX}$
R2^3SDF [16]	N-1	2N-1	$2log_2N$	$0.67log_4N$-1 (87.5%)	$0.67log_4N$-1	1/N	2N-1	$T_M + 2T_A + T_{MUX}$
R2^4SDF [16]	N-1	2N-1	$2log_2N$	$0.5log_4N$-1 (93.75%)	$0.5log_4N$-1	1/N	2N-1	$T_M + 2T_A + T_{MUX}$
Proposed	1.5N+1.5×log_2N-1.5	2N+1.5×log_2N-1.5	log_2N+1	log_4N-0.5 (100%)	-	1/N	2N+log_2N-1	$T_M + 2T_A + 3T_{MUX}$

Table Ⅳ COMPARISIONS OF TRANSISTOR REQUIREMENT AND LATENCY

Archi-cture	Components	Transi-stors	Late-ncy	Transistors * Latency
Chang [11]	1024 16-bit SRAMs 32 16-bit Adders 28 16-bit Multipliers	230748 (135%)	512	118142976 (133%)
Liu [12]	1048 16-bit SRAMs 32 16-bit Adders 28 16-bit Multipliers	233052 (136%)	524	122119248 (138%)
R2^2SDF [14]	1022 16-bit SRAMs 38 16-bit Adders 12 16-bit Multipliers	167138 (98%)	511	85407518 (96%)
R2SD^{2}F [21]	1192 16-bit SRAMs 22 16-bit Adders 12 16-bit Multipliers	175378 (103%)	591	103648398 (117%)
R2^3SDF [16]	1022 16-bit SRAMs 37.6 16-bit Adders 11.2 16-bit multipliers	163614 (96%)	511	83606754 (94%)
R2^4SDF [16]	1022 16-bit SRAMs 35.6 16-bit Adders 7.2 16-bit Multipliers	145992 (85%)	511	74601912 (84%)
Proposed	1045 16-bit SRAMs 25 16-bit Adders 14 16-bit Multipliers	171087 (100%)	519	88794153 (100%)

The R2^3SDF, using mixed radix architecture, roughly needs 2.8 general multipliers while R2^4SDF roughly needs 1.8 general multipliers [16].

and is more than the R2^3SDF and R2^4SDF. However, the high radix architecture can only process the FFT, whose size is a power of its high radix.

We also implement the proposed FFT architecture on the Xilinx Virtex-5 FPGA, XC5VSX240T-2 FF1738, and the corresponding place and route results for the different FFT sizes are shown in Table Ⅴ. Our results show that the proposed FFT architecture can achieve small area and high frequency.

Table Ⅴ AREA AND PERFORMANCE OF THE PROPOSED FFT ARCHITECTURES FOR 16 BITS

FFT Size	Area				Freq (MHz)	Latency (ns)
	LUTs	FFs	DSPs	BRAMs		
16	672	522	4	0	322	140
64	1110	752	8	0	303	498
256	1733	1073	12	0	297	1834
1024	2804	1589	16	3	298	7028
4096	8391	2780	20	4	295	27975

V. CONCLUSION

We propose a combined SDC-SDF pipelined FFT architecture which produces the output data in the normal order. The proposed *SDC PE* mainly reduces 50% complex multipliers, compared with the other radix-2 FFT designs. Therefore, the proposed FFT architecture is very attractive for the single-path pipelined radix-2 FFT processors with the input and output sequences in normal order.

REFERENCES

[1] L. J.Cimini, "Analysis and simulation of a digital mobile channel using orthogonal frequency division multiplexing," *IEEE Trans. Commun.*, 1985, pp. 665–675.

[2] J. M. Cioffi, *The commu*nications Handbook. Boca Raton, FL: CR, 1997

[3] Y.W.Lin, H.Y.Liu and C.Y.Lee, A 1-GS/s FFT/IFFT processor for UWB applications. IEEE J. Sol.-State Circ., pp.1726-1735, 2005.

[4] C. Cheng and K. K. Parhi, "High throughput VLSI architecture for FFT computation," IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 54, no. 10, pp. 339–344, Oct. 2007.

[5] 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.

[6] Y.Jung, H.Yoon, and J.Kim, "New efficient FFT algorithm and pipeline implementation results for OFDM/DMT applications," IEEE Trans. Consum. Electron., vol.49, pp.14-20, 2005.

[7] M. Shin and H. Lee, "A high-speed, four-parallel radix- 24 FFT processor for UWB applications," in Proc. IEEE ISCAS, 2008, pp. 960–963.

[8] J. H. McClellan and R. J. Purdy, "Applications of digital signal processing to radar," in Applications of Digital Signal Processing. Englewood Cliffs, NJ: Prentice-Hall, 1978, ch. 5.

[9] L.R.Rabiner, B.Gold, Theory and Application of Digital Signal Processing. Prentice-Hall, Inc., USA, pp.604-609, 1975.

[10] E.H.Wold, A.M.Despain, "Pipeline and parallel-pipeline FFT processors for VLSI implementation," IEEE Trans. Comput., c-33(5), 1984, pp. 414-426.

[11] Y.N.Chang, "An efficient VLSI architecture for normal I/O order pipeline FFT design," IEEE Trans. Circ. Syst. II:Exp. Briefs, 55(12), pp. 1234-1238, 2008.

[12] X. Liu, F.Yu, Z.K.Wang, "A pipelined architecture for normal I/O order FFT," J Zhejiang Univ-Sci C (Comput & Electron), 12(1), pp.76-82, 2011.

[13] G. Bi, E.V.Jones, "A pipelined FFT processor for word-sequential data," IEEE Trans. Acoust. Speech Signal Process, 37(12), pp. 1982-1985, 1989.

[14] S. He and M. Torkelson, "Designing pipeline FFT processor for OFDM (de)modulation," in Proc. URSI Int. Symp. Signals, Systems, and Electronics, vol. 29, pp. 257–262, Oct. 1998.

[15] T.Sansaloni, A.Perez-Pascual, V.Torres, and J.Valls, "Efficient pipeline FFT processors for WLAN MIMOOFDM systems," Electron. Lett, 2005, 41(19), pp.1043-1044.

[16] J.Y.Oh, M.S.Lim, "Area and Power Efficient Pipeline FFT Algorithm," Proc. IEEE Workshop on Signal Processing System Design and Implementation, 2005, pp.520-525.

[17] T.Cho, S.Tsai, and H. Lee, "A High-Speed Low-Complexity Modified Radix-2^5 FFT Processor for High Rate WPAN Applications," IEEE Trans. Very Large Scale Integr.(VLSI) Syst., vol. 21, no.1, pp. 187-191, Jan. 2013.

[18] A. M. Despain, "Fourier transform computer using CORDIC iterations," IEEE Trans. Comput., vol. C-23, no. 10, pp. 993–1001, Oct. 1974.

[19] A.Cortes, I.Velez, J.F. Sevillano, "Radix rk FFTs: matricial representation and SDC/SDF pipeline implementation," IEEE Trans. Signal Process., 2009, 57(7), pp.2824-2839.

[20] M.Garrido, J.Grajal, M.Sanchez and O. Gustafsson, "Pipelined Radix-2^k Feedforward FFT Architectures," IEEE Trans. Very Large Scale Integr.(VLSI) Syst., vol. 21, no.4, pp. 23-32, Jan. 2013.

[21] L. Yang, K. Zhang, H. Liu, J. Huang, and S. Huang, "An efficient locally pipelined FFT processor," IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 53, no. 7, pp. 585–589, Jul. 2006.

[22] T.Lenart, and V. Owall, "Architectures for Dynamic Data Scaling in 2/4/8K Pipeline FFT Cores," IEEE Trans. Very Large Scale Integr.(VLSI) Syst., vol. 14, no. 11, pp. 1286-1290, Nov. 2006.

[23] K.Yang, S.Tsai, and G. Chuang, "MDC FFT/IFFT Processor With Variable Length for MIMO-OFDM Systems," IEEE Trans. Very Large Scale Integr.(VLSI) Syst., vol. 21, no.4, pp. 720-731, Apr. 2013.

[24] M.Ayinala, M.Brown, and K. Parhi, "Pipelined Parallel FFT Architectures via Folding Transformation," IEEE Trans. Very Large Scale Integr.(VLSI) Syst., vol. 20, no. 6, pp. 1068-1081, Jun. 2012.

[25] N. H. E. Weste and D. Harris, CMOS VLSI Design: A Circuits and Systems Perspective. Boston, MA: Pearson/Addison-Wesley, 2005.

[26] B. Gold, and C. M. Rader, "Digital Processing of Signal," New York:McGraw-Hill Book Co., 1969, ch. 6.