Journal of Zhejiang University-SCIENCE C (Computers & Electronics)
ISSN 1869-1951 (Print); ISSN 1869-196X (Online)
www.zju.edu.cn/jzus; www.springerlink.com
E-mail: jzus@zju.edu.cn

A pipelined architecture for normal I/O order FFT

Xue LIU[†], Feng YU[†‡], Ze-ke WANG

(*Department of Instrument Engineering, Zhejiang University, Hangzhou 310027, China*)

[†]E-mail: liuxue0412@tom.com; osfengyu@zju.edu.cn

Received July 2, 2010; Revision accepted Oct. 11, 2010; Crosschecked Dec. 8, 2010

Abstract: We present a novel pipelined fast Fourier transform (FFT) architecture which is capable of producing the output sequence in normal order. A single-path delay commutator processing element (SDC PE) has been proposed for the first time. It saves a complex adder compared with the typical radix-2 butterfly unit. The new pipelined architecture can be built using the proposed processing element. The proposed architecture can lead to 100% hardware utilization and 50% reduction in the overall number of adders required in the conventional pipelined FFT designs. In order to produce the output sequence in normal order, we also present a bit reverser, which can achieve a 50% reduction in memory usage.

Key words: Fast Fourier transform (FFT), Single-path delay commutator (SDC), Pipelined FFT, Bit reverser
doi:10.1631/jzus.C1000234 **Document code:** A **CLC number:** TN91

1 Introduction

Fast Fourier transform (FFT) has played a significant role in digital signal processing applications, especially in recent years. It has been widely used in many advanced communication systems such as asymmetric digital subscriber line (ADSL), wireless networks, digital video broadcasting (DVB), ultra wide band (UWB), and digital audio broadcasting (DAB) (Sung *et al*., 2010). All these systems require FFT be computed in real time. Thus, improving the real-time processing capabilities is an important issue in designing a dedicated FFT circuit. Pipelined FFT architectures appear to be the primary solution to this issue due to low latency, high throughput, low power consumption, and small area (Garrido *et al*., 2009).

According to the literature, the pipelined FFT architectures can be classified into the following three types: multi-path delay commutator (MDC), single-path delay feedback (SDF), and single-path delay commutator (SDC). MDC architecture is used typically to process multiple-input data streams because of its high throughput rate (Lin *et al*., 2005; Cheng and Parhi, 2007). However, this approach is generally ill-suited to the single-input data stream. The major reason for this is its low hardware utilization. SDF architecture is the preferred solution to the single-input data stream, because the memory size required reaches the minimum and the multipliers are fully utilized (He and Torkelson, 1996; Yeh and Jen, 2003). However, the utilization of adders is still very low. SDC architecture (Bi and Jones, 1989; Cortes *et al*., 2009) is seldom used to process the single-input data stream, because it uses more memory resources than SDF and has a more complicated control. Note that the output orders of these pipelined architectures are bit-reversed and they need to be reversed to normal order. Chang (2008) presented a pipelined architecture which can reduce the required number of adders by half and generate the output sequence in normal order. The main disadvantage of this architecture is that, it breaks the data integrity and increases the implementation complexity of the computational units as the input sequence is divided into two parallel half-word sequences.

In this paper, we propose a novel pipelined FFT architecture, which can produce the output sequence

‡ Corresponding author

in normal order. Compared with the conventional pipelined architectures, the proposed architecture reduces the hardware requirement for adders and the implementation complexity of hardware using the proposed single-path delay commutator processing element (SDC PE).

2 Review of pipelined FFT architecture

The N-point discrete Fourier transform (DFT) is defined by

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

Here, $W_N=\mathrm{e}^{-2\pi \mathrm{j}/N}$, and N is any integer power of two. The radix-2 algorithm can be deduced from DFT by separating the N-point DFT into many related 2-point DFTs. The data flow graph (DFG) of 16-point radix-2 FFT is shown in Fig. 1.

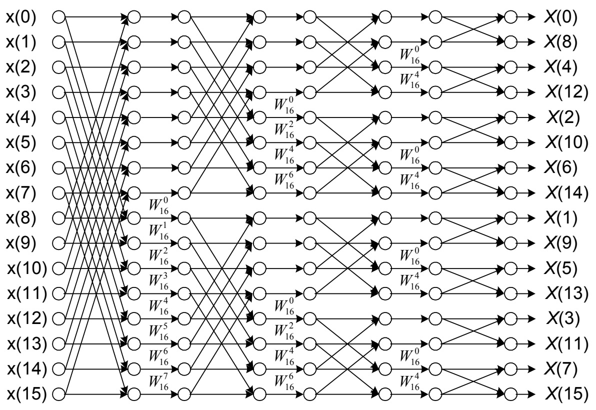

Fig. 1 Data flow graph of 16-point radix-2 FFT

Assuming that data are introduced serially into the FFT processor, the corresponding MDC and SDF architectures can be deduced from the DFG of the FFT algorithm. The radix-2 MDC architecture (Rabiner and Gold, 1975), as shown in Fig. 2a, is the most direct implementation approach of pipelined FFT. Its hardware utilization is only 50%. The radix-2 SDF architecture (Wold and Despain, 1984), as shown in Fig. 2b, reduces the memory requirements with respect to the radix-2 MDC architecture. The memory and multipliers in the radix-2 SDF architecture are fully utilized, but the utilization of adders is still 50%. In Fig. 2, the outputs produced by the SDF and MDC architectures are in bit-reversed order, so a block, called the bit reverser, should be included to reverse the output order to normal order.

Besides the basic radix-2 approaches, various higher radix pipelined FFT architectures have been proposed over the past several decades. They are radix-4 MDC (Swartzlander *et al.*, 1984), radix-4 SDC (Bi and Jones, 1989), radix-4 SDF (Despain, 1974), etc. In general, the hardware utilization of radix-r MDC designs is $1/r$ unless there are some special constraints on the input data stream (Cheng and Parhi, 2007). If r is a power of two, radix-r SDC and SDF can be realized by radix-2^k SDF, such as radix-2^3 SDF (Sansaloni *et al.*, 2005) and radix-2^4 SDF (Oh and Lim, 2005). The radix-2^k (k>1) SDF has a similar DFG to radix-2 SDF. The main difference between radix-2^k (k>1) SDF and radix-2 SDF is the number of complex multipliers. In radix-2^k (k>1) SDF architectures, some complex multipliers can be replaced by constant multipliers.

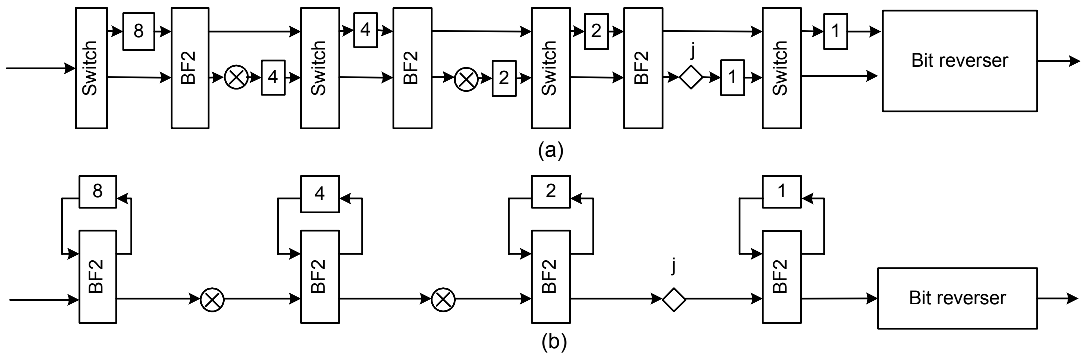

Fig. 2 The conventional pipelined architectures: (a) Radix-2 MDC FFT (N=16); (b) Radix-2 SDF FFT (N=16)
MDC: multi-path delay commutator; SDF: single-path delay feedback

3 The proposed pipelined FFT architecture

Aiming at the single-input data stream, we propose a new SDC pipelined architecture, which saves 50% of the required number of adders compared with the conventional pipelined architectures. However, this gain comes at the expense of increased memory size and latency. By optimizing the implementation of the bit reverser, the total buffer size of the proposed architecture is reduced. It is close to that of the conventional SDF architectures when processing large-point FFT.

3.1 The proposed single-path delay commutator architecture

In order to improve the hardware utilization of adders of the SDF pipelined design, we propose a new SDC pipelined architecture (Fig. 3), which consists of processing elements (PEs), complex multipliers, a butterfly unit, data buffers, and a bit reverser. The control unit and coefficient memory are omitted for clarity. Table 1 shows the data permutation order of each stage for 16-point FFT.

We propose a new type of PE, called the single-path delay commutator processing element (SDC PE), which reduces one real-number adder and one real-number subtracter compared with the typical radix-2 butterfly unit. BF2 used in the last stage is identical to that in radix-2 SDF. As shown in Fig. 4, the SDC PE consists of an arithmetic unit (called half-butterfly) and three data commutators (called data commutators 1, 2, 3). The buffer size of SDC PE

Table 1 Data permutation order of each stage for 16-point FFT based on our SDC design

Cycle	Permutation order			
	Stage 1	Stage 2	Stage 3	Stage 4
0	0_r, 0_i	–	–	–
1	8_r, 8_i	–	–	–
2	2_r, 2_i	–	–	–
3	10_r, 10_i	–	–	–
4	4_r, 4_i	–	–	–
5	12_r, 12_i	–	–	–
6	6_r, 6_i	0_r, 0_i	–	–
7	14_r, 14_i	4_r, 4_i	–	–
8	1_r, 1_i	2_r, 2_i	–	–
9	9_r, 9_i	6_r, 6_i	–	–
10	3_r, 3_i	8_r, 8_i	0_r, 0_i	–
11	11_r, 11_i	12_r, 12_i	2_r, 2_i	–
12	5_r, 5_i	10_r, 10_i	4_r, 4_i	–
13	13_r, 13_i	14_r, 14_i	6_r, 6_i	–
14	7_r, 7_i	1_r, 1_i	8_r, 8_i	–
15	15_r, 15_i	5_r, 5_i	10_r, 10_i	–
16	–	3_r, 3_i	12_r, 12_i	–
17	–	7_r, 7_i	14_r, 14_i	–
18	–	9_r, 9_i	1_r, 1_i	0_r, 0_i
19	–	13_r, 13_i	3_r, 3_i	2_r, 2_i
…	…	…	…	…
25	–	–	15_r, 15_i	14_r, 14_i
26	–	–	–	1_r, 1_i
27	–	–	–	3_r, 3_i
…	…	…	…	…
33	–	–	–	15_r, 15_i

*m*_r and *m*_i (*m*=0, 1, …, 15) represent the real and imaginary parts of the intermediate computed results with index *m*, respectively. The indexes of the data generated at each stage correspond to their locations in the data flow graph shown in Fig. 1

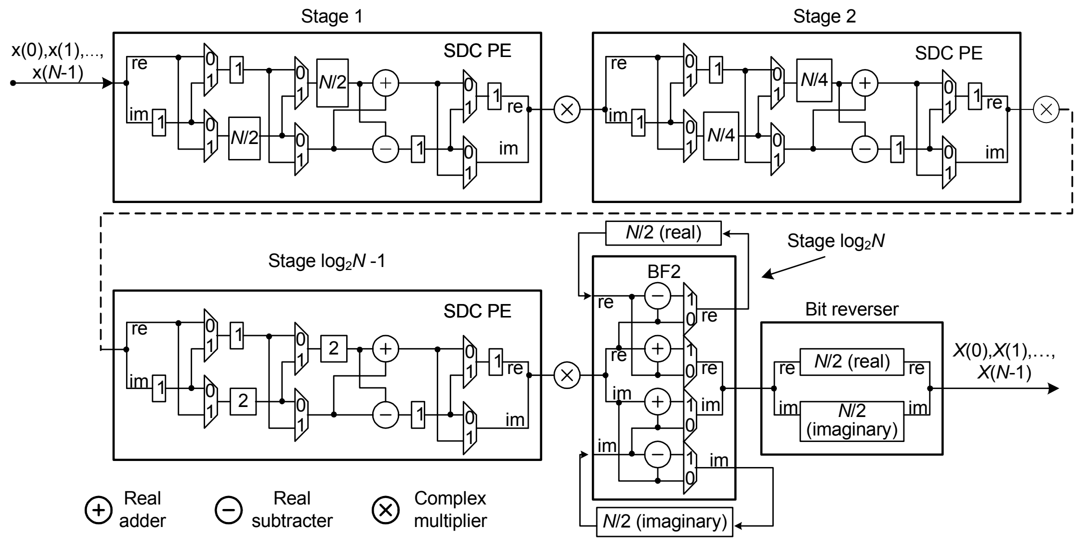

Fig. 3 The proposed single-path delay commutator (SDC) pipelined architecture

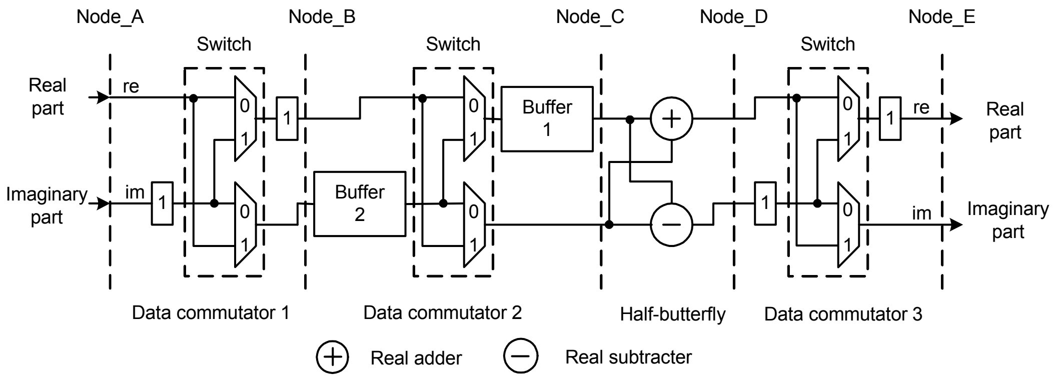

Fig. 4 Single-path delay commutator processing element (SDC PE) structure

is $N/2^S+2$, where S is the index of the stage. Table 2 shows the detailed data processing order of SDC PE located in the first stage for 16-point FFT. First, the input data are shuffled by data commutator 1 to generate two data sequences that comprise the even- and odd-indexed data. The two data sequences are ordered again so that their 'distance' is just $N/2^S$. A snapshot of the reordering process in data commutator 2 is captured in Fig. 5. From cycle 9 to cycle 16, the reordered outputs consist of the output of buffer 1 and the upper input of node B. In the next 8 cycles, they consist of the outputs of buffers 1 and 2. The reordered outputs then enter the computational unit called half-butterfly. The half-butterfly is fully utilized and the number of adders required is only one half of that required in the typical radix-2 butterfly unit. Finally, the computational results shuffled by data commutator 3 are exported in the complex format.

Besides the PEs, the internal data permutation order is the other major difference between our design and the conventional SDF architectures. The conventional SDF architectures export the computational data at each stage based on the top-down order in Fig. 1. As illustrated in Table 1, our proposed architecture forms a completely new output order after the first stage of the SDC PE. It consists of all the even-indexed datasets followed by the ones with an odd index. The BF2 in the last stage operates on the data pair with successive indexes so that a data buffer of size $N/2$ is required. The total data buffer size of the proposed architecture (excluding the bit reverser) is $1.5N+(\log_2 N-2)\times 2$, which is 50% more than that of the conventional SDF architecture. Reduction of the total data buffer size of our architecture is the next problem to solve.

Table 2 Data processing order of SDC PE located in the first stage of the proposed architecture for 16-point FFT for nodes A–E

Cycle	Processing order			
	A (input)	B	C/D (add/sub)	E (output)
0	0_r, 0_i	–	–	–
1	1_r, 1_i	0_r, 1_r	–	–
2	2_r, 2_i	0_i, 1_i	–	–
3	3_r, 3_i	2_r, 3_r	–	–
4	4_r, 4_i	2_i, 3_i	–	–
…	…	…	…	…
8	8_r, 8_i	6_i, 7_i	–	–
9	9_r, 9_i	8_r, 9_r	0_r, 8_r	–
10	10_r, 10_i	8_i, 9_i	0_i, 8_i	0_r, 0_i
11	11_r, 11_i	10_r, 11_r	2_r, 10_r	8_r, 8_i
12	12_r, 12_i	10_i, 11_i	2_i, 10_i	2_r, 2_i
…	…	…	…	…
17	–	–	1_r, 9_r	14_r, 14_i
18	–	–	1_i, 9_i	1_r, 1_i
19	–	–	3_r, 11_r	9_r, 9_i
…	…	…	…	…
25	–	–	–	15_r, 15_i

m_r and m_i (m=0, 1, …, 15) represent the real and imaginary parts of the intermediate computed results with index m, respectively. The indexes of the data generated at each stage correspond to their locations in the data flow graph shown in Fig. 1

3.2 Implementation of normal-order output

Note that the pipelined FFT design produces the output data in bit-reversed order. In order to generate normal-order outputs, an extra bit reverser that reorders the entire N-point output data is needed. The direct implementation of the bit reverser requires a data buffer of size $2N$ because of the differences of the read and write addresses. The size of the data buffer used in the bit reverser can be reduced to N by using a modified addressing method (Chang, 2008). Using this method, the datasets with an odd index are written

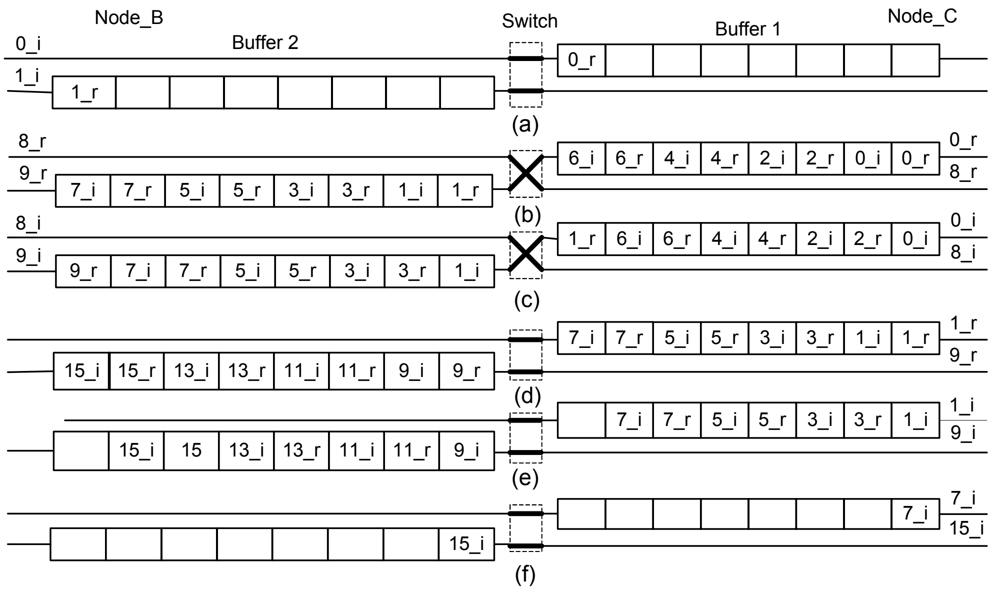

Fig. 5 Data commutator 2 operations: (a) cycle 2; (b) cycle 9; (c) cycle 10; (d) cycle 17; (e) cycle 18; (f) cycle 24

into memory in normal order and retrieved from memory in bit-reversed order, while the ones with an even index are written into memory in bit-reversed order and retrieved from memory in normal order.

Applying the above method to our design, the size of the data buffer used in the bit reverser can be reduced to $N/2$ because of the special output order of our design. Therefore, the total data buffer size of the proposed architecture is $2N+(\log_2 N-2)\times 2$, which is close to those of the conventional SDF architectures when processing large-point FFT. The structure of the bit reverser shown in Fig. 6 is very simple. It consists of a write address generator, a read address generator, and two $N/2$ dual-port RAMs. Each RAM is used to store the real and imaginary parts, respectively.

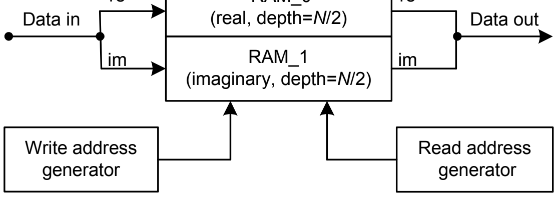

Fig. 6 Bit reverser structure

The normal output order $\{0, 1, \ldots, m, \ldots, N-1\}$ can be represented by $\{\mathrm{BR}_L(0), \mathrm{BR}_L(1), \ldots, \mathrm{BR}_L(m), \ldots, \mathrm{BR}_L(N-1)\}$, where the function $\mathrm{BR}_L(m)$ represents the L-bit bit-reversed representation of index m. For N-point FFT, L is set to $\log_2 N$. Assuming that m is expressed by the binary number $\{m_{L-1}, m_{L-2}, \ldots, m_1, m_0\}_2$, the function $\mathrm{BR}_L(m)$ is then given by $\{m_0, m_1, \ldots, m_{L-2}, m_{L-1}\}_2$. As mentioned before, the output sequences produced by our FFT design are the even-indexed datasets followed by the ones with an odd index. The bit-reversed representation of the successive even and odd indexes can be expressed by

$$\begin{cases} \mathrm{BR}_L(m)=\{0, m_1, \ldots, m_{L-2}, m_{L-1}\}_2, & \text{if } m \text{ is even}, \\ \mathrm{BR}_L(m)=\{1, m_1, \ldots, m_{L-2}, m_{L-1}\}_2, & \text{if } m \text{ is odd}. \end{cases} \tag{2}$$

Eq. (2) indicates that the output order of our FFT design consists of all the data in the range $(0, N/2-1)$ followed by the data in the range $(N/2, N-1)$. According to this output order, we reduce the buffer size required to $N/2$. The detailed operation of the bit reverser is described by the following four steps:

1. In the first $N/2$ cycles, the even-indexed datasets are written in bit-reversed order. The write address is $\{0, m_1, \ldots, m_{L-2}, m_{L-1}\}_2$.

2. In the next $N/2$ cycles, the even-indexed datasets are retrieved from memory in normal order and the odd-indexed datasets are written into memory in normal order. The read and write addresses are $\{m_{L-1}, m_{L-2}, \ldots, m_1, 0\}_2$ and $\{m_{L-1}, m_{L-2}, \ldots, m_1, 1\}_2$, respectively.

3. In the next $N/2$ cycles, the odd-indexed datasets are retrieved in bit-reversed order and the even-indexed datasets of the next outputs are written

in bit-reversed order. The read and write addresses are $\{1, m_1, \ldots, m_{L-2}, m_{L-1}\}_2$ and $\{0, m_1, \ldots, m_{L-2}, m_{L-1}\}_2$, respectively.

4. Return to the second step.

Through the simple shift operation, the read and write addresses in the second and third steps can be mapped to the same location in the range $(0, N/2-1)$, so a data buffer of size $N/2$ is sufficient for the bit reverser. Table 3 illustrates the detailed data reordering scheduling evolved from Table 1.

4 Comparison and analysis

Table 4 presents the hardware requirement of our design and other pipelined architectures based on the basic radix-2 algorithm. It first lists the memory requirement. The column labeled 'internal buffer' shows the internal buffer size without regard to the output order. Apparently, the typical SDF design reaches the minimum requirement for the internal buffer. The column labeled 'overall buffer' shows the total buffer size when the bit reverser is included to reverse the output order to normal order. The buffer size of the proposed bit reverser is reduced to $N/2$, so the total buffer size of the proposed architecture is equal to $2N+(\log_2N-2)\times2$, which is close to that of the typical SDF design when processing large-point FFT. Compared with the typical SDF design, the increased memory size of our design is less than 1% when N is greater than 1024. Note that the internal buffer size of the parallel MDC design is equal to N (Cheng and Parhi, 2007). However, this size depends on the

Table 4 Hardware requirement comparison for radix-2 pipelined designs

Architecture	Internal buffer[#]	Overall buffer[##]	Adder	Multiplier
CPD				
SDF	$N-1$	$2N-1$	$2\log_2N$	$\log_2N-1$
SDC	$2N-2$	$3N-2$	$2\log_2N$	$\log_2N-1$
MDC	$1.5N-2$	$2.5N-2$	$2\log_2N$	$2\log_2N-2$
PMDC[*]	N	$2N$	$2\log_2N$	$2\log_2N-2$
MMDC[**]	$1.5N$	$2N$	$\log_2N+1$	$\log_2N-1$
PSDC	$1.5N+(\log_2N-2)\times2$	$2N+(\log_2N-2)\times2$	$\log_2N+1$	$\log_2N-1$

[*] Cheng and Parhi (2007); [**] Chang (2008). [#] Internal buffer size without regard to the output order; [##] Total buffer size when the bit reverser is included to reverse the output order to normal order. CPD: conventional pipelined designs; SDF: single-path delay feedback; SDC: single-path delay commutator; MDC: multi-path delay commutator; PMDC: parallel MDC; MMDC: modified MDC; PSDC: proposed SDC

Table 3 Data reordering scheduling of the bit reverser for 16-point FFT

Cycle	Stage 4	RAM_0_1 value[*]								Output	
	m_r_i	0	1	2	3	4	5	6	7	m_r_i	BR$_4(m)$
0	0_r_i	–	–	–	–	–	–	–	–	–	–
…	…	…	…	…	…	…	…	…	…	…	…
8	1_r_i	0_r_i	8_r_i	4_r_i	12_r_i	2_r_i	10_r_i	6_r_i	14_r_i	–	–
9	3_r_i	1_r_i	8_r_i	4_r_i	12_r_i	2_r_i	10_r_i	6_r_i	14_r_i	0_r_i	0
10	5_r_i	1_r_i	3_r_i	4_r_i	12_r_i	2_r_i	10_r_i	6_r_i	14_r_i	8_r_i	1
11	7_r_i	1_r_i	3_r_i	5_r_i	12_r_i	2_r_i	10_r_i	6_r_i	14_r_i	4_r_i	2
12	9_r_i	1_r_i	3_r_i	5_r_i	7_r_i	2_r_i	10_r_i	6_r_i	14_r_i	12_r_i	3
13	11_r_i	1_r_i	3_r_i	5_r_i	7_r_i	9_r_i	10_r_i	6_r_i	14_r_i	2_r_i	4
14	13_r_i	1_r_i	3_r_i	5_r_i	7_r_i	9_r_i	11_r_i	6_r_i	14_r_i	10_r_i	5
15	15_r_i	1_r_i	3_r_i	5_r_i	7_r_i	9_r_i	11_r_i	13_r_i	14_r_i	6_r_i	6
16	0^*_r_i	1_r_i	3_r_i	5_r_i	7_r_i	9_r_i	11_r_i	13_r_i	15_r_i	14_r_i	7
17	2^*_r_i	0^*_r_i	3_r_i	5_r_i	7_r_i	9_r_i	11_r_i	13_r_i	15_r_i	1_r_i	8
18	4^*_r_i	0^*_r_i	3_r_i	5_r_i	7_r_i	2^*_r_i	11_r_i	13_r_i	15_r_i	9_r_i	9
19	6^*_r_i	0^*_r_i	3_r_i	4^*_r_i	7_r_i	2^*_r_i	11_r_i	13_r_i	15_r_i	5_r_i	10
20	8^*_r_i	0^*_r_i	3_r_i	4^*_r_i	7_r_i	2^*_r_i	11_r_i	6^*_r_i	15_r_i	13_r_i	11
21	10^*_r_i	0^*_r_i	8^*_r_i	4^*_r_i	7_r_i	2^*_r_i	11_r_i	6^*_r_i	15_r_i	3_r_i	12
22	12^*_r_i	0^*_r_i	8^*_r_i	4^*_r_i	7_r_i	2^*_r_i	10^*_r_i	6^*_r_i	15_r_i	11_r_i	13
23	14^*_r_i	0^*_r_i	8^*_r_i	4^*_r_i	12^*_r_i	2^*_r_i	10^*_r_i	6^*_r_i	15_r_i	7_r_i	14
24	1^*_r_i	0^*_r_i	8^*_r_i	4^*_r_i	12^*_r_i	2^*_r_i	10^*_r_i	6^*_r_i	14^*_r_i	15_r_i	15
25	3^*_r_i	1^*_r_i	8^*_r_i	4^*_r_i	12^*_r_i	2^*_r_i	10^*_r_i	6^*_r_i	14^*_r_i	0^*_r_i	0^*

[*] RAM_0_1 address depth=8. m_r_i (m=0, 1, …, 15) represents the real and imaginary parts of the data with index m; m^* represents the data with index m of the next output datasets. The last column shows that the output sequence order generated by our proposed architecture is converted back to normal order

condition that well-aligned parallel input data are available for the first-stage butterfly unit. However, this condition cannot always exist.

Table 4 also lists the required numbers of computational units for various radix-2 pipelined designs. We see that our architecture and the modified MDC design (Chang, 2008) save almost 50% of the number of adders required compared with other designs. For example, when N equals 4096, our proposed architecture can save 22 real-number adders. Moreover, our design requires the same number of multipliers as the typical SDC/SDF design. Using the aforementioned radix-2^k (k>1) SDF approaches, such as radix-2^3 and radix-2^4, can further reduce the number of multipliers required. Note that our design can also be applied to the radix-2^k SDF pipelined designs. In Table 4, we list only the hardware requirement for the radix-2 pipelined design.

Table 4 shows that the hardware requirement of our design is close to that of the modified MDC design when processing large-point FFT. By dividing the input sequence into two parallel half-word sequences, the modified MDC design reaches the minimum requirement for both the adder and memory. However, the division of the input data stream breaks the data integrity so that the complex adder and complex multiplier should be designed either by time-multiplexing the shared arithmetic units or by using a digit-serial method (Chang, 2008). This undoubtedly increases the implementation complexity of the computational units. Similarly, the modified MDC design is ill-suited to the computation of floating-point FFT. Our design will not suffer these limitations, because the proposed SDC PE not only keeps the data integrity, but also reduces the implementation complexity of the computational units.

5 Conclusions

We propose a novel pipelined FFT architecture which produces the output data in normal order. Using the proposed single-path delay commutator processing element (SDC PE), the new pipelined architecture achieves a reduction in adders' usage of up to 50%. In order to produce the output sequence in normal order, a bit reverser saving 50% of memory requirement is proposed. Compared with the conventional pipelined FFT designs, the proposed architecture reduces the number of adders and hardware implementation complexity. Therefore, the proposed FFT architecture is suitable for the implementation of pipelined FFT processors with the input and output sequence in normal order.

References

Bi, G., Jones, E.V., 1989. A pipelined FFT processor for word-sequential data. *IEEE Trans. Acoust. Speech Signal Process.*, **37**(12):1982-1985. [doi:10.1109/29.45545]

Chang, Y.N., 2008. An efficient VLSI architecture for normal I/O order pipeline FFT design. *IEEE Trans. Circ. Syst. II: Exp. Briefs*, **55**(12):1234-1238. [doi:10.1109/TCSII.2008.2008074]

Cheng, C., Parhi, K.K., 2007. High-throughput VLSI architecture for FFT computation. *IEEE Trans. Circ. Syst. II: Exp. Briefs*, **54**(10):863-867. [doi:10.1109/TCSII.2007.901635]

Cortes, A., Velez, I., Sevillano, J.F., 2009. Radix r^k FFTs: matricial representation and SDC/SDF pipeline implementation. *IEEE Trans. Signal Process.*, **57**(7):2824-2839. [doi:10.1109/TSP.2009.2016276]

Despain, A.M., 1974. Fourier transform computer using CORDIC iterations. *IEEE Trans. Comput.*, **c-23**(10):993-1001. [doi:10.1109/T-C.1974.223800]

Garrido, M., Parhi, K.K., Grajal, J., 2009. A pipelined FFT architecture for real-valued signals. *IEEE Trans. Circ. Syst. I: Reg. Papers*, **56**(12):2634-2643. [doi:10.1109/TCSI.2009.2017125]

He, S., Torkelson, M., 1996. A New Approach to Pipeline FFT Processor. Proc. 10th Int. Symp. on Parallel Processing, p.766-770. [doi:10.1109/IPPS.1996.508145]

Lin, Y.W., Liu, H.Y., Lee, C.Y., 2005. A 1-GS/s FFT/IFFT processor for UWB applications. *IEEE J. Sol.-State Circ.*, **40**(8):1726-1735. [doi:10.1109/JSSC.2005.852007]

Oh, J.Y., Lim, M.S., 2005. Area and Power Efficient Pipeline FFT Algorithm. Proc. IEEE Workshop on Signal Processing System Design and Implementation, p.520-525. [doi:10.1109/SIPS.2005.1579923]

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

Sansaloni, T., Perez-Pascual, A., Torres, V., Valls, J., 2005. Efficient pipeline FFT processors for WLAN MIMO-OFDM systems. *Electron. Lett.*, **41**(19):1043-1044. [doi:10.1049/el:20052597]

Sung, T.Y., Hsin, H.C., Cheng, Y.P., 2010. Low-power and high-speed CORDIC-based split-radix FFT processor for OFDM systems. *Dig. Signal Process.*, **20**(2):511-527. [doi:10.1016/j.dsp.2009.08.008]

Swartzlander, E.E., Young, W.K.W., Joseph, S.J., 1984. A radix 4 delay commutator for fast Fourier transforms processor implementation. *IEEE J. Sol.-State Circ.*, **19**(5):702-709. [doi:10.1109/JSSC.1984.1052211]

Wold, E.H., Despain, A.M., 1984. Pipeline and parallel-pipeline FFT processors for VLSI implementation. *IEEE Trans. Comput.*, **c-33**(5):414-426. [doi:10.1109/TC.1984.1676458]

Yeh, W.C., Jen, C.W., 2003. High-speed and low-power split-radix FFT. *IEEE Trans. Signal Process.*, **51**(3):864-874. [doi:10.1109/TSP.2002.806904]