An Efficient VLSI Architecture for Normal I/O Order Pipeline FFT Design

Yun-Nan Chang, *Member, IEEE*

***Abstract*—In this paper, an efficient VLSI architecture of a pipeline fast Fourier transform (FFT) processor capable of producing the normal output order sequence is presented. A new FFT design based on the decimated dual-path delay feed-forward data commutator unit by splitting the input stream into two half-word streams is first proposed. The resulting architecture can achieve full hardware efficiency such that the required number of adders can be reduced by half. Next, in order to generate the normal output order sequence, this paper also presents a sequence conversion method by integrating the conversion function into the last-stage data commutator module.**

***Index Terms*—Fast Fourier transform (FFT), pipeline FFT.**

I. Introduction

THE fast Fourier transform (FFT) algorithm has been widely used in many discrete-time signal processing systems. Especially in recent years, many advanced communication systems including digital audio broadcasting (DAB), digital video broadcasting (DVB), wireless networks, or high-speed digital subscriber line (ADSL) systems all require a core FFT module to process the orthogonal frequency division multiplexing (OFDM) function. Therefore, how to design an efficient dedicated FFT circuit especially for the emerging OFDM applications is a very important issue. One of the popular FFT design approaches that has been frequently adopted in the past is called pipeline FFT design [1]–[12]. This approach is especially favored for high-throughput real-time FFT applications because it allocates the individual dedicated data-path for each FFT processing stage to achieve a high level of parallel processing. Depending on the data shuffling method, the pipeline FFT architectures can be further divided into two categories. The first category of FFT designs are built on the so-called multipath delay feed-forward (MDF) data commutator [1]–[5]. However, this approach will suffer the large internal data buffer as well as low functional unit utilization unless multiple-input data streams are available. For the single-input data stream, another class of pipeline FFT designs, called single-delay feedback (SDF) FFT [6]–[11], which circulates the partial intermediate computation results generated by each stage of data-path back to the same stage in order to reuse its data buffer will be preferred. Not only the overall size of internal data buffer can be reduced, but the multiplier units can also be fully utilized. Unfortunately, the utilization of the adder units is still very low. Both MDF and SDF FFT designs can process the continuous input data stream online such that they are well suitable for OFDM applications. However, the transformed sequence generated by these pipeline processors will be in the bit-reverse order which has to be converted back to normal by an additional sequence converter.

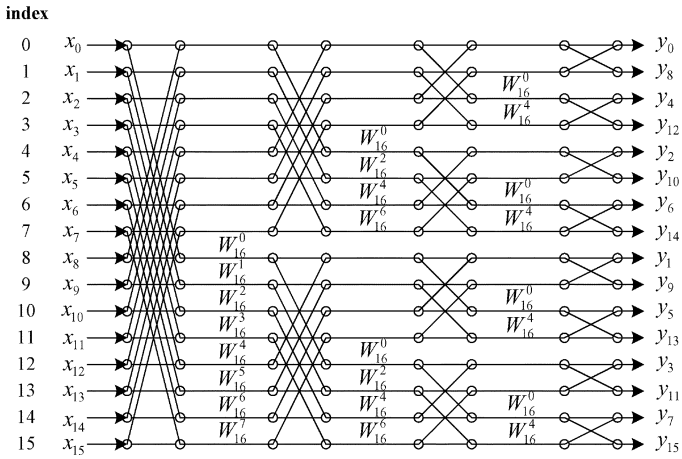

Fig. 1. Data-flow graph of the 16-point in-place *decimation-in-frequency* FFT computation.

This paper aims to propose an efficient pipeline FFT architecture that can generate the normal output order sequence. The remainder of this paper is organized as follows. Section II reviews the design of the conventional pipeline FFT architectures. The proposed FFT design is discussed in detail in Section III. Our design can overcome the problem of low hardware utilization of butterfly units. How to avoid the buffer overhead due to this new design will also be addressed in the same section. Section IV presents some comparison results with the traditional pipeline approaches. Finally, some conclusions are given in Section V.

II. Review of Pipeline FFT Processor Architecture

The discrete Fourier transform (DFT) y_k of an N-point sequence x_n is expressed as follows:

$$y_k = \sum_{n=0}^{N-1} x_n W_N^{nk}, \quad k = 0 \ldots N-1. \tag{1}$$

The coefficient W_N is equal to $e^{-(2\pi j/N)}$ and is often referred to as the twiddle factor. Instead of direct implementation of the computationally intensive DFT, the FFT algorithm is used to

Manuscript received April 14, 2008; revised June 16, 2008 and August 02, 2008. Current version published December 12, 2008. This paper was recommended by Associate Editor Z. Wang.

The author is with the Department of Computer Science and Engineering, National Sun Yat-sen University, Kaohsiung 80424, Taiwan (e-mail: ynchang@cse.nsysu.edu.tw).

Digital Object Identifier 10.1109/TCSII.2008.2008074

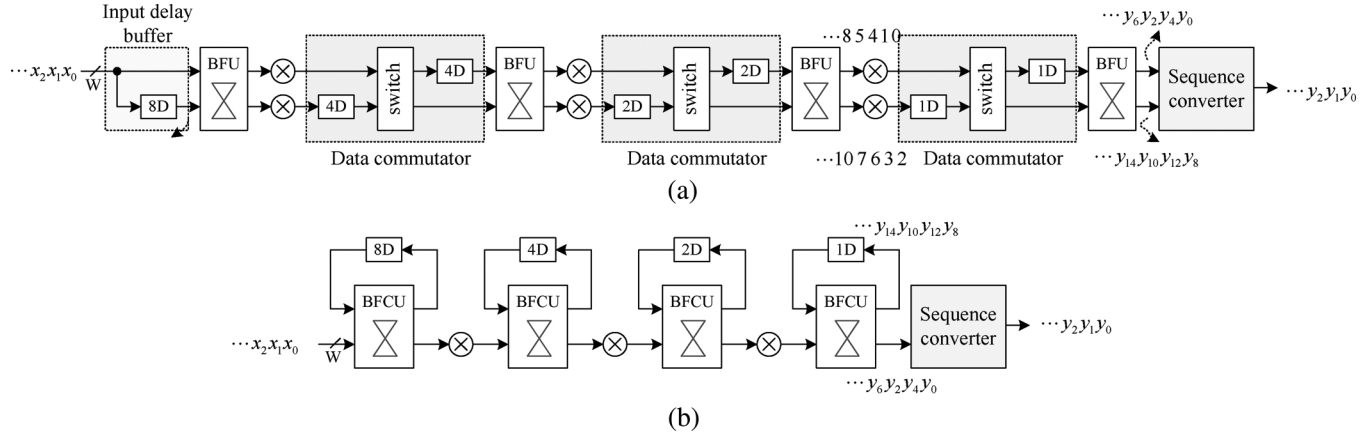

Fig. 2. Block diagram of the sequential I/O 16-point pipeline FFT architectures for (a) MDF FFT and (b) SDF FFT designs.

factorize a large-point DFT recursively into many small-point DFTs such that the overall operations involved can be drastically reduced. In this paper, we assume N is the power of two. Fig. 1 shows the resulted data flow graph (DFG) of a 16-point DFT computation based on the radix-2 decomposition. Assuming that the input data enters the FFT processor serially, the processor circuit can be designed by mapping each stage of the FFT computation sequentially into the corresponding data-path as shown in Fig. 2. For the MDF design, unless well-aligned parallel input data streams are available, as described in [2], an input delay buffer stage will be required to buffer the first half points of data. The buffering of the inputs will lead to only 50% hardware utilization, and the total internal buffer size of $W \times (3N/2)$ for N-point FFT, where W represents the word-length of the data. In order to reduce the buffer size for the single-input stream, the SDF radix-2 FFT design shown in Fig. 2(b) will store one of the butterfly unit outputs back to the buffer of the same stage such that the overall buffer size can be reduced to $W \times N$. The output sequences generated by both pipeline FFT architectures shown in Fig. 2 will follow a special bit-reverse order; therefore, a block labelled as *sequence converter* has to be included to convert the sequence back to normal.

In addition to the direct radix-2 approach, there have been various radix-2 FFT decomposition methods proposed before including radix-2^2 [6], radix-2^3 [9], and so on. These various algorithms will lead to the similar FFT DFGs with only the difference on the distribution of those twiddle factors. By merging the nontrivial factors into fewer stages, some multipliers shown in Fig. 2 can be either removed or realized by simple dedicated constant multipliers. Since the simplification of the multiplier is not the focus of this paper, the general form of pipeline structure in Fig. 2 will be used. In addition, although higher radix of FFT algorithms have also been addressed before [2], [3], [8], they will not be considered in this paper because they often lead to worse hardware utilization of pipeline FFT designs unless there some special conditions exist [2].

III. Proposed New Pipeline FFT Architecture

As mentioned in the previous section, the MDF FFT is suitable for multiple-input data streams targeted for very high-throughput FFT applications. On the other hand, the SDF FFT is favored for the single-input data stream due to the more efficient use of the internal buffer. Targeted on the single-input stream, this paper will first propose a modified MDF FFT which can use fewer functional units than the SDF FFT does. Next, by integrating the function of sequence converter and some data commutator, the total size of the data buffer used by our design will be the same as SDF FFT.

A. Design of Dual Half-Word Serial MDF FFT Architecture

In order to overcome the problem of low hardware utilization for the pipeline MDF FFT design with single data stream, the paper proposes a new modified MDF N-point FFT architecture shown in Fig. 3. Here, the incoming data will be divided into the most significant half and least-significant half words. Next, the two half words will be shuffled by a preprocessing commutator stage to form two data streams that consist of the input data with odd and even index, respectively. Table I illustrates the data permutation result by the data commutator of each stage for an example of 16-point FFT computation. Here, we assume one pipeline stage is inserted to the data path between every two commutators. The notation k_L and k_H represent the least significant half and most significant half of the intermediate computation data with index k, assuming that the data produced by each stage are indexed according to its position in the DFG shown in Fig. 1. After the preprocessing stage, the data will enter the typical radix-2 MDF FFT architecture with the following two main exceptions. First, the width of the data stream of the upper and lower rails of this architecture is $W/2$ which is the half of the original input data word-length. In other words, it will take two cycles for the butterfly unit and the twiddle factor multipliers to process the computation of one entire data. Therefore, these functional units can be designed by either digit-serial approach or sharing the common arithmetic units in time-multiplexing approach such that the cost of these functional units will be only about half of those used in typical pipeline FFT designs shown in Fig. 2. These functional units can be fully utilized such that the overall number of adders required will be equivalent to only half of the radix SDF FFT.

The other major difference of the proposed FFT architecture compared with the typical MDF FFT shown in Fig. 2(a) is the internal data processing order. The typical MDF FFT architectures process each stage of computation according to the top-down order of the butterfly operation shown in the in-place FFT DFG

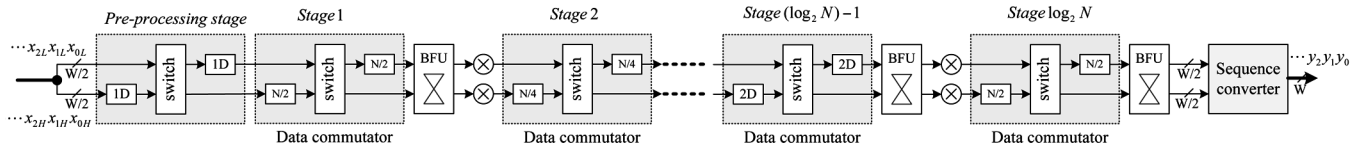

Fig. 3. Block diagram of the proposed new FFT architecture.

TABLE I
DATA OUTPUT ORDER OF EACH COMMUTATOR STAGE FOR THE 16-POINT DFT BASED ON THE PROPOSED FFT ARCHITECTURE

Stage / Cycle	I/P	Pre-pro--cessing	1	2	3	4
0	0	-	-	-	-	-
1	1	0_L,1_L	-	-	-	-
2	2	0_H,1_H	-	-	-	-
3	3	2_L,3_L	-	-	-	-
4	4	2_H,3_H	-	-	-	-
...	...	...	...	...	...	...
8	8	6_H,7_H	-	-	-	-
9	9	8_L,9_L	0_L,8_L	-	-	-
10	10	8_H,9_H	0_H,8_H	-	-	-
11	11	10_L,11_L	2_L,10_L	-	-	-
12	12	10_H,11_H	2_H,10_H	-	-	-
13	13	12_L,13_L	4_L,12_L	-	-	-
14	14	12_H,13_H	4_H,12_H	0_L,4_L	-	-
15	15	14_L,15_L	6_L,14_L	0_H,4_H	-	-
16	-	14_H,15_H	6_H,14_H	2_L,6_L	-	-
17	-	-	1_L,9_L	2_H,6_H	0_L,2_L	-
18	-	-	1_H,9_H	8_L,12_L	0_H,2_H	-
19	-	-	3_L,11_L	8_H,12_H	4_L,6_L	-
...	...	...	...	...	...	...
25	-	-	-	3_H,7_H	1_L,3_L	-
26	-	-	-	9_L,13_L	1_H,3_H	0_L,1_L
27	-	-	-	9_H,13_H	5_L,7_L	0_H,1_H

such as Fig. 1. However, in our proposed design, after the permutation of the first stage data commutator, the incoming data sequence order has been altered to be a new one which consists of the all the even-indexed data followed by the odd-indexed ones. Therefore, for the last stage of the butterfly unit which operates on the data pair with successive index, another big data commutator providing $N/2$-point data shuffling is required. Although excluding the final sequence converter, the overall data buffer size used for all the data commutators of the proposed architecture is $(W/2)\times 3N$ which is the same as the conventional MDF FFT, it is still 50% more than the SDF FFT design. How to further utilize these extra data buffer is the next issue to solve.

B. Generation of Normal Output Order

The pipeline FFT design is suitable for the process of the continuous input data stream with the normal sequential order. However, the output order of transformed data produced by the general pipeline FFT design is not sequential. It follows a so-called bit-reversed order. To reverse the output sequence back to the normal, an extra sequence converter that can shuffle the order of the entire N-point outputs has to be included. The direct implementation of the sequence converter will require one double-buffer consisting of two memory banks as illustrated in Fig. 4(a). Each memory block can hold N-point data. Suppose the transformed data y_i is written to the memory location i according to its index. Since the output order of the general pipeline FFT designs is bit-reverse, the memory write access order can be represented by $br_{LN}(0), br_{LN}(1), br_{LN}(2), \ldots,$

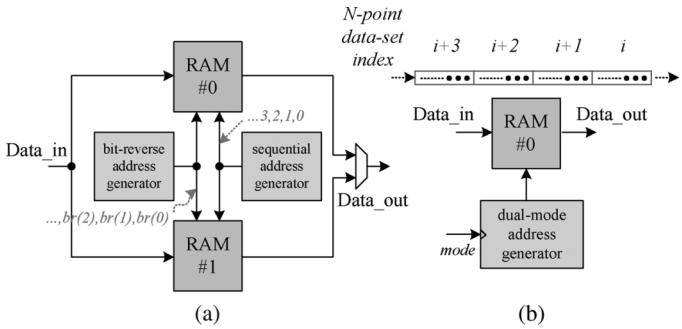

Fig. 4. Block diagram of the sequence converter.

where the function $br_{LN}(i)$ represents the value of the LN-bit bit-reverse representation of integer i. For example, $br_4(13)$ and $br_5(13)$ are equal to 11 and 22 respectively. For N-point FFT, LN will be set to $log_2 N$. After storing the entire N-point transformed data in one memory bank, they can be fetched out following the sequential address order to constitute a normal-order sequence. The double-buffer circuit is required due to the discrepancy of the memory read and memory write access orders. However, it can be observed that if we store the output y_i to the memory location $br_{LN}(i)$ instead of i, they can also be retrieved following the bit-reverse access order to form a normal-order sequence. Therefore, in order to alleviate the need of the double-buffer, the alternative access methods have to be adopted for different N-point data sets. As shown in Fig. 4(b), the even-indexed data set are written in the bit-reverse and fetched in the normal order while the odd-indexed one are written in the normal and fetched in the bit-reverse order. This addressing approach can make the memory location which is being just fetched to be immediately available for the current input data.

By alternatively changing the addressing mode, the data buffer can be used more efficiently such that only one memory block of size N can be sufficient for the sequence conversion. Therefore, the entire buffer size of the overall FFT architecture shown in Fig. 3 will be $W \times (5N/2)$. However, recall that, in the proposed architecture, the last-stage butterfly unit requires a large data commutator to collect pairs of data with successive index for the butterfly operation. In order to reduce the overall buffer size, this paper proposes a new order conversion circuit by relocating the sequence conversion block back into the input side of the last-stage butterfly unit such that it can be integrated with the function of the last-stage data commutator. Fig. 5 shows the modified block diagram for the last-stage butterfly unit and the associated data commutator. The butterfly execution order for the last stage has been modified in order to directly produce the normal-order transformed outputs. Table II shows the partial detailed data reordering schedule evolved

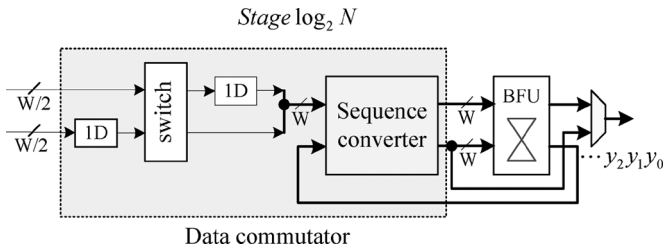

Fig. 5. Modified block diagram for the integration of the last butterfly unit and sequence converter.

from Table I for the 16-point DFT. It can be found from this table that an internal small data commutator first converts the dual-input half-word data streams back to a single-input full-word data stream. The notation m^* represents the data with index m of the next data set. The new sequence converter will first buffer one entire 16-point intermediate data set. Next, according to the desired output order $y_0, y_1, y_2, \ldots$, the associated butterfly inputs will be fetched. Different from the original proposed FFT architecture shown in Fig. 3, the last butterfly unit will process one complete butterfly operation each cycle instead of every two cycles. For the N-point FFT computation, the butterfly unit will be active for consecutive $N/2$ cycles to generate all the transformed data. The first half of the transformed sequence $y_0, y_1, \ldots y_{(N/2)-1}$ will be output directly. The rest of the transformed data will be written back to the internal buffer of the converter, and retrieved later during the next $N/2$ cycles.

Similar to the sequence converter design issue mentioned before, the proposed last-stage data commutator shown in Fig. 5 has to be carefully designed to avoid the use of the double-buffer. For the even N-point data set, the input data have to be written to the memory according to its index. For the odd set, the data are written to the special bit-reverse-mapped memory location. The data storage rule can be expressed in the following equation:

$$\begin{array}{ll} \mathrm{Mem}[i] = D_j(i), & \text{if j is even} \\ \mathrm{Mem}\left[2br_{LN1}\left(\left\lfloor \frac{i}{2} \right\rfloor\right) + i_{\mathrm{mod}2}\right] = D_j[i], & \text{if j is odd.} \end{array} \quad (2)$$

Here, the notation $\mathrm{Mem}[i]$ represents the memory content of address i. The notation $D_j(i)$ represents the intermediate input data with index i for the jth data set. The parameter $LN1$ is equal to $\log_2 N - 1$. The detailed operation of the last-stage butterfly unit with the associated data commutator function can be described by the pseudocode shown in Fig. 6. The function $MR(k)$ represents the memory read operation that provides the memory content at the address k. The function $MW(k, m)$ represents the memory write operation that will store the data m into the memory location k. The function $BF(d1, d2)$ represents the radix-2 butterfly operation of input data $d1$ and $d2$ and returns two computation results. The variable t represents the tth cycle for the last butterfly unit in processing the N-point data-set stored inside the converter. Based on the data storage rule expressed in (2), and the data access operation shown in Fig. 6, one block of memory with size $N \times W$ can be feasible to realize both functions of the sequence conversion and data commutator. The total size required for the original separate designs is $3N/2 \times W$.

TABLE II
DETAILED DATA PROCESSING ORDER BASED ON THE INTEGRATED DATA COMMUTATOR AND SEQUENCE CONVERTER DEIGN

Stage	Stage 3 O/P	Converter I/P	Converter O/P	Butterfly operation	Output data
...	...	...	...	...	...
17	$0_L, 2_L$	-	-	-	-
18	$0_H, 2_H$	0,-	-	-	-
19	$4_L, 6_L$	2,-	-	-	-
...	...	...	...	...	...
23	$12_L, 14_L$	10,-	-	-	-
24	$12_H, 14_H$	12,-	-	-	-
25	$1_L, 3_L$	14,-	-	-	-
26	$1_H, 3_H$	1,-	-	-	-
27	$5_L, 7_L$	3,-	-	-	-
...	...	...	...	...	...
31	$13_L, 15_L$	11,-	-	idle	-
32	$13_H, 15_H$	13,-	-	idle	-
33	$0^*_L, 2^*_L$	15,-	-	idle	-
34	$0^*_H, 2^*_H$	$0^*, y_8$	0,1	$\surd$	y_0
35	$4^*_L, 6^*_L$	$2^*, y_9$	8,9	$\surd$	y_1
36	$4^*_H, 6^*_H$	$4^*, y_{10}$	4,5	$\surd$	y_2
37	$8^*_L, 10^*_L$	$6^*, y_{11}$	12,13	$\surd$	y_3
38	$8^*_H, 10^*_H$	$8^*, y_{12}$	2,3	$\surd$	y_4
39	$12^*_L, 14^*_L$	$10^*, y_{13}$	10,11	$\surd$	y_5
40	$12^*_H, 14^*_H$	$12^*, y_{14}$	6,7	$\surd$	y_6
41	$1^*_L, 3^*_L$	$14^*, y_{15}$	14,15	$\surd$	y_7
42	$1^*_H, 3^*_H$	1^*,-	y_8	idle	y_8
43	$5^*_L, 7^*_L$	3^*,-	y_9	idle	y_9
44	$5^*_H, 7^*_H$	5^*,-	y_{10}	idle	y_{10}

```
for (t = 0; t < N; t++){
  if (t < N/2){
    if (j_mod2){  // odd data set
      addr1 = 2br_LN1(t);
      addr2 = 2br_LN1(t) + 1;}
    else{        //even data set
      addr1 = 2t;
      addr2 = 2t + 1;}
    data1 = MR(addr1);
    data2 = MR(addr2);
    (out, y(t + N/2)) = BF(data1, data2);
    MW(addr1, D_j(2t));
    MW(addr2, y(t + N/2));}
  else{
    if (j_mod2) //odd data set
      addr1 = 2br_LN1(t - N/2) + 1;
    else        //even data set
      addr1 = 2(t - N/2) + 1;
    out = MR(addr1);
    MW(addr1, D_j(2t - N + 1));}
}
```

Fig. 6. Pseudocode of the operation of the last-stage data commutator and the butterfly unit.

IV. COMPARISON RESULT

This section presents some comparison results of the proposed FFT architecture with other pipeline designs based on

TABLE III
EQUIVALENT NUMBER OF BUFFER WORDS AND ARITHMETIC UNITS REQUIRED FOR VARIOUS RADIX-2 PIPELINE FFT DESIGNS

Architecture	SDF	MDF	Parallel MDF [2]	Proposed
Internal buffer	N	$1.5N$	N	$1.5N$
Overall buffer	$2N$	$2.5N$	$2N$	$2N$
Adder	$2logN$	$2logN$	$2logN$	$logN$+1
Multiplier	$logN-1$	$2logN-2$	$2logN-2$	$logN-1$

the same radix-2 FFT algorithm. Table III first compares the required size of buffer. The first row shows the size of internal buffer only irrespective of the output sequence order. As expected, the SDF FFT design can lead to the most efficient use of the internal buffer such that it requires the least amount of the internal storage space. The second row further lists the total buffer required if the sequence converter is also taken into account in order to generate the normal output-order sequence. Since our proposed design has merged the order converter circuit with the last-stage commutator, it needs the same amount of buffer size as the SDF FFT design. It should be noted that the parallel MDF design described in [2] assumes the internal buffer size of N-word is used; however, this size is based on the assumption that two input data for each cycle can arrive in the exact order for the first-stage butterfly processing without the need of the extra input delay buffer as shown in Fig. 2. However, this assumption cannot always hold true.

Table III further compares the number of arithmetic units required for various radix-2 architectures. It can be found that the proposed architecture only requires almost half the number of adders compared with others. This saving on adders can represent a large reduction of the gate counts of the FFT circuit. For example, if N is equal to 8192, our design can lead to the reduction of 24 real-number adders, and these adders are typically realized by the fast adder architectures such as carry-lookahead adders. As for the demand of multipliers, in general, MDF designs will require double the number of multipliers compared with SDF, as illustrated by Fig. 2. However, since the multipliers used in our proposed architecture shown in Fig. 3 process one multiplication every two cycles, those two half-word multipliers between every two butterfly stages can actually be realized by a single full-word multiplier in the time-multiplexing approach. Therefore, the number of multipliers required by the proposed design and the SDF will be the same. It also should be noted that the number of multipliers can be further reduced by other variant radix-2 FFT approaches such as radix-2^2 and radix-2^3. They can substitute some of the general multipliers by simpler constant multipliers built by some adders and multiplexors. Since these techniques can be applied equivalent to all the pipeline designs, only the basic radix-2 comparison result is provided in this paper.

V. CONCLUSION

This paper proposed a novel VLSI architecture of the pipeline FFT processor. By rearranging the input data sequence into dual half-word data streams, the new pipeline FFT architecture can be obtained by using the modified feed-forward data commutators. The resulted architecture can save half number of adders compared with the other pipeline FFT designs. In order to provide a transformed output sequence with normal order, a novel sequence conversion mechanism is also proposed by integrating the converter with the last-stage data commutator such that the overall buffer size required is the minimum. Therefore, the proposed FFT design methodology can lead to very efficient pipeline FFT circuit suitable for the processing of the normal input and output order sequence.

REFERENCES

[1] G. Bi and E. Jones, "A pipelined FFT processor for word-sequential data," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 37, no. 12, pp. 1982–1985, Dec. 1989.
[2] 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. 863–867, Oct. 2007.
[3] E. E. Swartzlander, W. K. W. Young, and S. J. Joseph, "A radix-4 delay commutator for fast Fourier transform processor implementation," *IEEE J. Solid-State Circuits*, vol. SC-19, no. 5, pp. 702–709, Oct. 1984.
[4] E. Bidet, D. Castelain, C. Joanblanq, and P. Senn, "A fast single-chip implementation of 8192 complex point FFT," *IEEE J. Solid-State Circuits*, vol. 30, no. 3, pp. 300–305, Mar. 1995.
[5] S. H. Park, D. H. Kim, D. S. Han, K. S. Lee, S. J. Park, and J. R. Choi, "Sequential design of a 8192 complex point FFT in OFDM receiver," in *Proc. IEEE Asia-Pacific Conf. Adv. Syst. Integr. Circuits*, Seoul, Korea, Aug. 1999, pp. 262–265.
[6] S. He and M. Torkelson, "Designing pipeline FFT processor for OFDM (de)modulation," in *Proc. URSI Int. Symp. Signals, Syst., Electron.*, Sep. 1998, pp. 257–262.
[7] T. Lenart and V. Owall, "A 2048 complex point FFT processor using a novel data scaling approach," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Bangkok, Thailand, May 2003, pp. 45–48.
[8] C.-C. Wang, J.-M. Huang, and H.-C. Cheng, "A 2 k/8 k mode small-area FFT processor for OFDM demodulation of DVB-T receivers," *IEEE Trans. Consumer Electron.*, vol. 51, no. 1, pp. 28–32, Feb. 2005.
[9] L. Jia, Y. Gao, J. Isoaho, and H. Tenhunen, "A new VLSI-oriented FFT algorithm and implementation," in *Proc. 11th Annu. IEEE Int. Conf. Adv. Syst. Integr. Circuits*, Rochester, NY, Sep. 1998, pp. 337–341.
[10] W.-C. Yeh and C.-W. Jen, "High-speed and low-power split-radix FFT," *IEEE Trans. Signal Process.*, vol. 51, no. 3, pp. 864–874, Mar. 2003.
[11] S. Y. Park, N. I. Cho, S. U. Lee, K. Kim, and J. Oh, "Design of 2 k/4 k/8 k-point FFT processor based on CORDIC algorithm in OFDM receiver," in *Proc. IEEE Pacific Rim Conf. Commun., Comput. Signal process.*, Victoria, BC, Canada, Aug. 2001, vol. 2, pp. 457–460.
[12] J.-Y. Oh and M.-S. Lim, "Fast Fourier transform processor based on low-power and area-efficient algorithm," in *Proc. IEEE Asia-Pacific Conf. Adv. Syst. Integr. Circuits*, Fukuoka, Japan, Aug. 2004, pp. 198–201.