[17] L. Dadda, "On serial-input multipliers for two's complement numbers," *IEEE Trans. Computers*, vol. 38, Sept. 1989.
[18] P. Ienne and M. A. Viredaz, "Bit-serial multipliers and squarers," *IEEE Trans. Computers*, vol. 43, Dec. 1994.
[19] P. Ingelhan, B. Jonsson, B. Sikstrom, and L. Wanhammar, "A high-speed bit-serial processing element," in *Proc. Eur. Conf. on Circuit Theory and Design (ECCTD'89)*, 1989, pp. 162–165.
[20] S. G. Smith, "Serial/parallel automultiplier," *Electron. Lett.*, vol. 23, no. 8, pp. 413–415, 1987.
[21] J. Valls, T. Sansaloni, M. M. Peiró, and E. Boemo, "Fast FPGA-based pipelined digit-serial/parallel multipliers," in *IEEE Int. Symp. on Circuits and Systems (ISCAS'99)*, vol. 1, Orlando, FL, 1999, pp. 482–485.
[22] R. Gnanasekaran, "A fast serial-parallel binary multiplier," *IEEE Trans. Computers*, vol. C-34, Aug. 1985.

# An Efficient Pipelined FFT Architecture

Yun-Nan Chang and Keshab K. Parhi

***Abstract*—This paper presents an efficient VLSI architecture of the pipeline fast Fourier transform (FFT) processor based on radix-4 decimation-in-time algorithm with the use of digit-serial arithmetic units. By combining both the feedforward and feedback commutator schemes, the proposed architecture can not only achieve nearly 100% hardware utilization, but also require much less memory compared with the previous digit-serial FFT processors. Furthermore, in FFT processors, several modules of ROM are required for the storage of twiddle factors. By exploiting the redundancy of the factors, the overall ROM size can be effectively reduced by a factor of 2.**

***Index Terms*—Digit-serial, fast Fourier transform (FFT), pipelined FFT, radix-4 FFT.**

## I. Introduction

The fast Fourier transform (FFT) plays an important role in the design and implementation of discrete-time signal processing algorithms and systems. In recent years, motivated by the emerging applications in the modern digital communication systems and television terrestrial broadcasting systems, there has been tremendous growth in the design of high-performance dedicated FFT processors [1], [2]. Pipelined FFT processor is a class of real-time FFT architectures characterized by continuous processing of the input data which, for the reason of the transmission economy, usually arrives in the word sequential format. However, the FFT operation is very communication intensive which calls for spatially global interconnection. Therefore, much effort on the design of FFT processors focuses on how to efficiently map the FFT algorithm to the hardware to accommodate the serial input for computation. This paper presents a novel FFT implementation based on the use of digit-serial arithmetic which can lead to very efficient architectures.

Manuscript received August 15, 2000; revised February 12, 2003. This paper was recommended by Associate Editor Y. Wang.

Y.-N. Chang is with the Department of Computer Science and Engineering, National Sun Yat-Sen University, Kaohsiung, Taiwan 804, R.O.C. (e-mail: ynchang@cse.nsysu.edu.tw).

K. K. Parhi is with the Department of Electrical and Computer Engineering, University of Minnesota, MN 55455 USA.

Digital Object Identifier 10.1109/TCSII.2003.811439

## II. Review of FFT Processors

The discrete Fourier transform (DFT) $X(k)$ of an $N$-point sequence $x(n)$ is defined by

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk}, \qquad k = 0, 1, \ldots, N-1$$
$$W_N = e^{-j(2\pi/N)}. \tag{1}$$

Instead of direct implementation of the computationally intensive DFT, the FFT algorithm is used to factorize a large point DFT recursively into many small point DFTs such that the overall operations involved can be drastically reduced. There are two well-known types of FFT algorithms called *decimation-in-time (DIT)* and *decimation-in-frequency (DIF)* FFT which can be derived from each other by transposition. For example, according to radix-4 *DIT* FFT, (1) can be decomposed and expressed in the matrix form as follows:

$$\left[X(k), X\left(k+\frac{N}{4}\right), X\left(k+\frac{N}{2}\right), X\left(k+\frac{3N}{4}\right)\right]^T = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} \begin{bmatrix} W_N^0 F_0(k) \\ W_N^k F_1(k) \\ W_N^{2k} F_2(k) \\ W_N^{3k} F_3(k) \end{bmatrix}. \tag{2}$$

Here

$$F_{n_1}(k) = \sum_{n_2=0}^{(N/4)-1} x(n_1 + 4n_2) W_{N/4}^{n_2 \cdot k}$$
$$\text{for } n_1 = 0, 1, 2, 3;\ k = 0, 1, \ldots, \frac{N}{4} - 1.$$

Radix-2 and radix-4 are the most common radices used in FFT decompositions. Radix-4 decomposition is more attractive since it requires less amount of multiplication operations for FFT and reduces the number of multiplications from $N^2$ for direct implementation of DFT to only $(\log_4 N - 1)N$.

Since the data sequence $x(n)$ arrives sequentially, the parallel data flow graph has to be projected along the order of input sequence in order to obtain efficient pipeline architectures. As (2) shows, each stage of FFT computation consists of retrieving the data $F_0(k)$, $F_1(k)$, $F_2(k)$, $F_3(k)$ for specific $k$, and the corresponding twiddle factor multiplication, followed by the multiplication of the radix-4 butterfly matrix. Direct implementation of (2) requires three multipliers to perform the twiddle factor multiplication as shown in Fig. 1(a) [1], [3]. Here, the commutator is used to generate the proper data sequence for the following twiddle factor multiplication by swapping/exchanging the output data coming from the previous stage. The salient feature of this feedforward approach is that the trivial factor $W_N^0 (=1)$ in the twiddle matrix can be reflected in the hardware. However, unless four input data are sampled in parallel, this architecture cannot achieve full efficiency. For most of the applications where FFT processor must be interfaced to a continuous word serial stream, it is only possible to achieve 25% hardware utilization as there is a $4:1$ mismatch between the bandwidth of input data rate and that of the processor. (In general, the utilization for radix-$r$ butterfly unit is $1/r$.) In order to compensate this mismatch, a fully utilized architecture based on the use of digit-serial arithmetic units has been proposed in [4].

The other way of implementing (2) is to use a single multiplier for the twiddle factor multiplication as shown in Fig. 1(b). Instead of generating the vector $[F_0(k), W_N^k F_1(k), W_N^{2k} F_2(k), W_N^{3k} F_3(k)]^T$ in parallel as shown in Fig. 1(a), this scheme generates each element

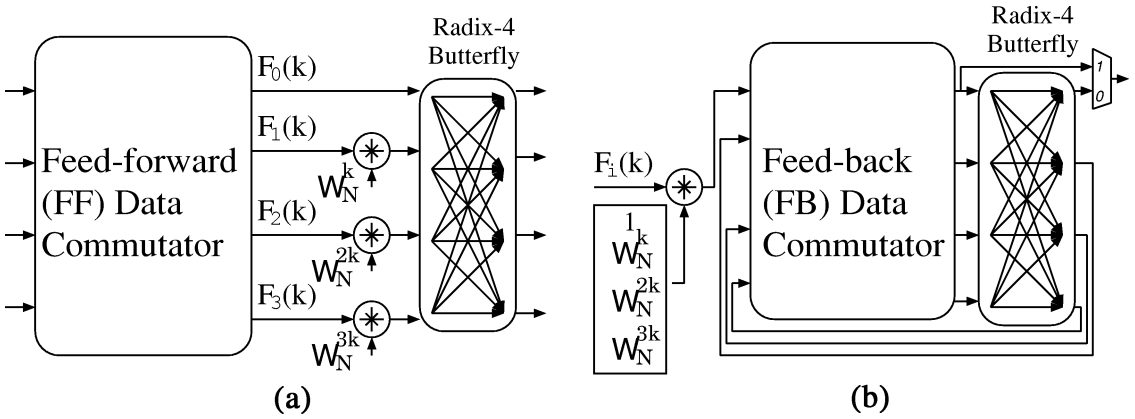


Fig. 1. Radix-4 commutator. (a) Feedforward scheme. (b) Feedback scheme.

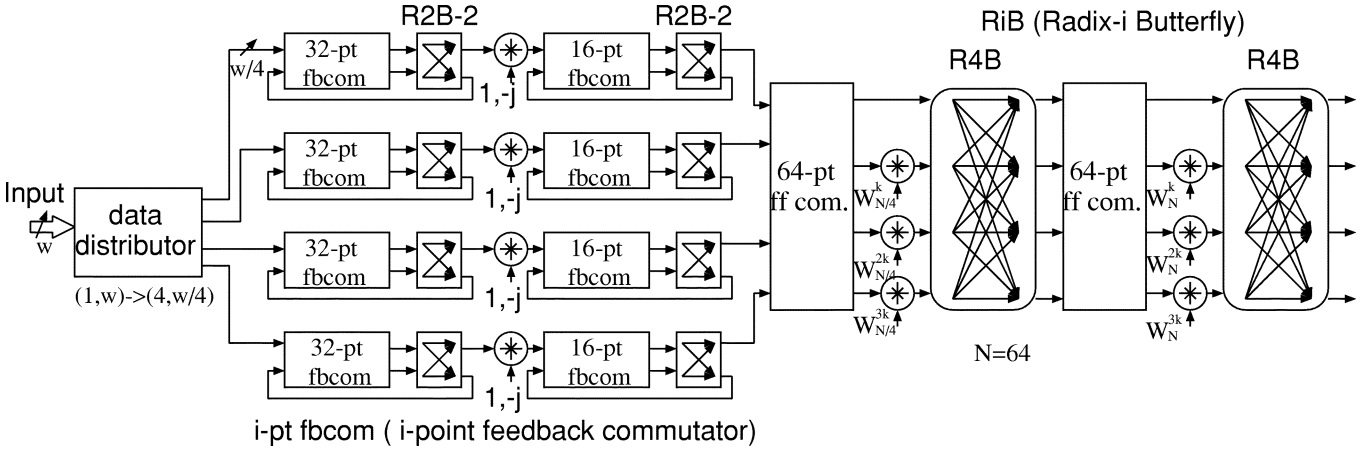


Fig. 2. 64 point FFT chip architecture.

in the vector sequentially by computing one factor multiplication each time. In order to do so, only one of the radix-4 butterfly outputs passes through the multiplier; the rest of them are fed back into the commutator. In addition, the commutator is placed next to the twiddle factor multiplier to generate the vector output in order. This implementation not only increases the utilization of multipliers, but it minimizes the required amount of memory by efficiently storing the butterfly outputs back to the registers inside the commutator. However, this scheme still suffers from low hardware utilization. Therefore, in [5], the radix-4 FFT algorithm was implemented using radix-2 FFT architecture such that the utilization of adders was increased from 25% to 50%. The multiplier utilization is equal to 75% since one of every four factor multiplications shown in Fig. 1(b) involves the trivial factor $W_N^0$.

## III. Radix-4 Digit-Serial FFT Processors

In general, the feedforward scheme can achieve better hardware utilization while the feedback scheme can lead to less memory requirement. By combining these two schemes, this paper proposes a novel FFT architecture which not only increases the hardware efficiency, but also incorporates the methodology of feedback scheme to reduce the memory requirement [6]. Fig. 2 shows the block diagram of the proposed FFT architecture for a transformation size of 64 based on DIT FFT because its factor access pattern can reduce the twiddle ROM access rate. The details of each functional block are shown in Fig. 3. The first FFT stage, based on the feedback scheme, consists of a data distributor followed by four parallel digit-serial feedback commutator and butterfly data paths. The data distributor serves as a format converter which converts input data format from bit-parallel into digit-serial. In the meantime, it also distributes the $N$-point input data equally into four parallel $N/4$-point digit-serial data streams. Following the distributor, each data stream then passes through the commutator in order to perform a radix-4 butterfly operation which, as shown in Fig. 2, is actually implemented by two radix-2 butterfly stages in order to reduce the number of adders. One of the radix-2 butterfly outputs in this stage will be fed back to the delay commutator which corresponds to the radix-2 version of feedback implementation scheme shown in Fig. 1(b). This approach results in a significant saving of the memory required. This approach is similar to [5] except that in our architecture, the data is operated in digit-serial format.

All the remaining stages after the first stage adopt the feedforward scheme similar to [4]. The detailed commutator architecture used for this scheme is shown in Fig. 3(a), where $N$-point commutator shuffles $N$-point data like matrix transposition to align the correct output data for the subsequent butterfly operations. The architecture of this commutator is composed of several shift-registers and a switch network. The radix-4 butterfly unit, as shown in Fig. 3(d) can be built by the interconnection of four radix-2 butterfly units. Since the first stage generates four digit-serial outputs continuously, the feedforward scheme in the remaining stages will not suffer from the problem of low utilization of arithmetic units. The digit-serial adders and multipliers can be fully utilized which is not possible for the pure feedback scheme.

In FFT architectures, the number of ROM modules required for the storage of factors depends on the commutator scheme being used. For the feedback scheme shown in Fig. 1(b), the number of ROM modules required is equal to $(\log_4 N - 1)$. For the feedforward approach, parallel multipliers are used such that more twiddle factors have to be provided at the same time. This can be implemented either by the use of more smaller ROM modules or unified ROM modules with large I/O width. The advantage of this scheme is that ROM access rate can be

TABLE I
COMPARISON OF PIPELINED FFT ARCHITECTURES (* BIT-PARALLEL, ** DIGIT-SERIAL)

| | Bi & Jones [7] | He & Torkelson [5] | Hui et al [4] | Proposed |
|---|---|---|---|---|
| Commutator scheme | feedfoward | feedback | feedforward | mixed |
| Complex adders | $8log_4N^*$ | $4log_4N^*$ | $12log_4N^{**}$ | $8(log_4N+1)^{**}$ |
| Complex multipliers | $log_4N-1^*$ | $log_4N-1^*$ | $3(log_4N-1)^{**}$ | $3(log_4N-1)^{**}$ |
| Data Memory | $2.75N$ | $N$ | $2.5N$ | $1.18N$ |
| Twiddle factor ROM | $N$ | $N$ | $N$ | $0.5N$ |
| Computational efficiency | add 50% mul 75% | add 50% mul 75% | add 100% mul 100% | add~100% mul 100% |

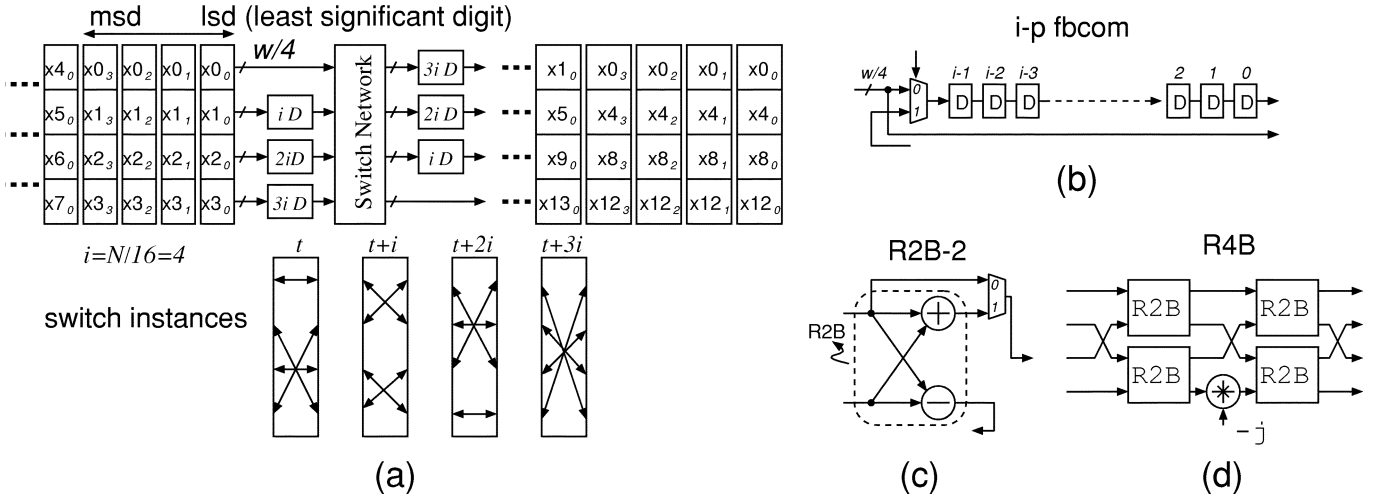


Fig. 3. (a) $N$-point ($N = 64$) FF commutator, (b) $i$-point feedback commutator, (c) radix-2 butterfly, and (d) radix-4 butterfly.

reduced by one fourth. In addition, if separate smaller ROM modules are used, their size can be further reduced by exploiting the individual redundancy characteristic of the twiddle factors stored in each module. This reduction can be explained by looking at ROM modules for those multipliers in the final stage shown in Fig. 2. Here, three ROMs in the final stage are responsible for providing the twiddle factors $W_N^k$, $W_N^{2k}$, and $W_N^{3k}$, respectively. The range of index $k$ is from 0 to $(N/4)-1$. The redundancy of the twiddle factors can be illustrated in (3) where the twiddle factors for $W_N^{ik}$ and $W_N^{i((N/4)-k)}$ share the same coefficient values. For example, $W_N^k$ and $W_N^{(N/4)-k}$ can be derived from each other by exchanging the real and imaginary parts. Therefore, by sharing the common contents, the size of ROM can be reduced by a factor of 2. This method can also be implemented for the feedback scheme, however, the additional control overhead may be large

$$\begin{array}{lll} & W_N^{ik} & W_N^{i((N/4)-k)} \\ i=1 & \cos\frac{2\pi k}{N}+j\sin\frac{2\pi k}{N} & \sin\frac{2\pi k}{N}+j\cos\frac{2\pi k}{N} \\ i=2 & \cos\frac{4\pi k}{N}+j\sin\frac{4\pi k}{N} & -\cos\frac{4\pi k}{N}+j\sin\frac{4\pi k}{N} \\ i=3 & \cos\frac{6\pi k}{N}+j\sin\frac{6\pi k}{N} & -\sin\frac{6\pi k}{N}+j\cos\frac{6\pi k}{N}. \end{array} \tag{3}$$

Since one radix-4 butterfly requires 8 complex adders and $N$-point FFT consists of $\log_4 N$ radix-4 stages, the total number of adders required in the proposed architecture equals $8\log_4 N$ plus extra 8 adders for the first stage butterfly operation. The number of storage elements required equals $N(1+(1/2))$ for the first feedback commutator plus

$$\frac{1}{4}N\left[\frac{3}{4}+\frac{3}{4}\left(1+\frac{1}{4}+\cdots+\frac{1}{4^{\log_4 N-3}}\right)\right]$$

for the rest of feedforward commutator. Although one feedback radix-4 stage is shown in Fig. 2, in fact, more feedforward stages can be replaced by feedback scheme. The tradeoff is that more feedback stages can reduce more memory but also increase more arithmetic units.

TABLE II
COMPARISON OF HIGH SPEED PARALLEL FFT ARCHITECTURES

| | He [5] | Swartzlander [1] | Proposed |
|---|---|---|---|
| Commutator | feedback | feedforward | mixed |
| No of adders | $16log_4N$ | $8log_4N$ | $8(log_4N+1)$ |
| No of multipliers | $4log_4N-1$ | $3(log_4N-1)$ | $3(log_4N-1)$ |
| Memory | $N$ | $2.5N$ | $1.18N$ |

## IV. DISCUSSION

Table I compares the hardware requirement for different FFT architectures. It is shown that the hardware utilization based on the digit-serial approach is higher than the bit-parallel implementation. Therefore, the required adder hardware of the proposed architecture is less than half compared with [5], [7] assuming the fast adder type is employed. The saving of multiplier hardware of the proposed architecture is not obvious since the implementation of digit-serial multipliers has some control overhead; however, the critical path of the digit-serial multiplier is smaller than that of the bit-parallel multiplier. Since the multiplication is often the critical operation in the complete FFT architecture, the digit-serial based FFT architecture can operate at faster speed. As for the data memory requirement, our proposed architecture needs slightly more data memory compared with [5], but the requirement is significantly lower compared with the other digit-serial based FFT processor [4]. Finally, for the twiddle factor ROM, compared with the bit-parallel FFT approach, our digit-serial architecture can not only reduce the access rate of ROM, but it is also suitable for the ROM reduction techniques.

Table II compares different FFT architectures for high speed applications assuming four input data arrive at a time. In this case, the corre-

sponding multipliers or butterfly units in Fig. 2 would be all in bit-parallel format. The proposed architecture achieves a significant savings of memory over [1], and requires less number of arithmetic units compared to [5].

In this paper, a novel design of radix-4 pipeline FFT has been discussed. The similar approach can also been extended to lower or higher radix FFT design. It can be applied to the radix-8 or radix-$2^3$ design which becomes more popular as it can take advantage of the simple implementation of the factor $W_8$ [8].

## REFERENCES

[1] 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, pp. 702–709, Oct. 1984.

[2] E. Bidet *et al.*, "A fast single-chip implementation of 8192 complex point FFT," *IEEE J. Solid-State Circuits*, vol. 30, pp. 300–305, Mar. 1995.

[3] B. Gold and T. Bially, "Parallelism in fast Fourier transform hardware," *IEEE Trans. Audio Electroacoust.*, vol. AU-21, pp. 5–16, 1973.

[4] C. C. W. Hui, T. J. Ding, and J. V. McCanny, "A 64-point Fourier transform chip for video motion compensation using phase correlation," *IEEE J. Solid-State Circuits*, vol. 31, pp. 1751–1761, Nov. 1996.

[5] S. He and M. Torkelson, "Design and implementation of a 1024-point pipeline FFT processor," in *Custom Integrated Circuits Conf.*, Santa Clara, CA, May 1998, pp. 131–134.

[6] Y.-N. Chang and K. K. Parhi, "Efficient FFT implementation using digit-serial arithmetic," in *Proc. 1999 IEEE Workshop on Signal Processing Systems*, Taipei, Taiwan, R.O.C., Oct. 1999.

[7] G. Bi and E. Jones, "A pipelined FFT processor for word-sequential data," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-37, pp. 1982–1985, Dec. 1989.

[8] L. Jia, Y. Gao, J. Isoaho, and H. Tenhunen, "A new VLSI-oriented FFT algorithm and implementation," in *Proc. 11th Annu. IEEE ASIC Conf.*, Rochester, NY, Sept. 1998, pp. 337–341.

# Corrections to "On the Exact Realization of LOG-Domain Elliptic Filters Using the Signal Flow Graph Approach"

Costas Psychalinos and Spiridon Vlassis

In the above paper [1], on page 774, reference [4] contained incorrect page numbers. The correction is given below.

[4] D. R. Frey, "Exponential state-space filters: A generic current-mode design strategy," *IEEE Trans. Circuits Syst. I*, vol. 43, no. 12,no. 1, pp. 34–42, Jan. 1996.

## REFERENCES

[1] C. Psychalinos and S. Vlassis, "On the exact realization of LOG-domain elliptic filters usingthe signal flow graph approach," *IEEE Trans. Circuits Syst. II*, vol. 49, no. 12, pp. 770–774, Dec. 2002.

Manuscript received May 1, 2003.

The authors are with the Physics Department, Electronics Laboratory, Aristotle University of Thessaloniki, GR-54124 Thessaloniki, Greece (e-mail: cpsychal@physics.auth.gr; svals@skiathos.physics.auth.gr).

Digital Object Identifier 10.1109/TCSII.2003.814778