Circuits Systems Signal Process
Vol. 18, No. 4, 1999, Pp. 377–393

Real-Time FFT Algorithm Applied to On-Line Spectral Analysis*

Pei-Chen Lo[1] *and Yu-Yun Lee*[1]

Abstract. On-line running spectral analysis is of considerable interest in many electrophysiological signals, such as the EEG (electroencephalograph). This paper presents a new method of implementing the fast Fourier transform (FFT) algorithm. Our "real-time FFT algorithm" efficiently utilizes computer time to perform the FFT computation while data acquisition proceeds so that local butterfly modules are built using the data points that are already available. The real-time FFT algorithm is developed using the decimation-in-time split-radix FFT (DIT sr-FFT) butterfly structure. In order to demonstate the synchronization ability of the proposed algorithm, the authors develop a method of evaluating the number of arithmetic operations that it requires. Both the derivation and the experimental result show that the real-time FFT algorithm is superior to the conventional whole-block FFT algorithm in synchronizing with the data acquisition process. Given that the FFT size $N = 2^r$, real-time implementation of the FFT algorithm requires only $2/r$ the computational time required by the whole-block FFT algorithm.

1. Introduction and motivation

In an EEG, the rhythmicity provides a means of quantitatively describing the EEG records [5]. The development of an FFT algorithm [4] has made the running spectral analysis of long-term EEG records practical. For example, the compressed spectra method [2] based on the short-time Fourier transform (STFT) provides a visual interpretation of the time-varying frequency characteristics. The STFT has been widely used in commercially available EEG machines to analyze the frequency spectra of multichannel EEGs. On-line processing of digital EEG signals is helpful for intensive-care monitoring and diagnostic reference in clinics. In many applications we very often require the real-time display of the EEG fre-

* Received May 21, 1998; revised October 29, 1998. This work was supported by the National Science Council of Taiwan, Republic of China, under grant NSC87-2213-E-009-128.

[1]Department of Electrical and Control Engineering, National Chiao Tung University, 1001 Ta Hsueh Road, Hsinchu, 30050 Taiwan, Republic of China. E-mail:pclo@cc.nctu.edu.tw

quency spectra. As a result of advances in digital technology and instrumentation, some EEG instruments have more than 100 recording channels [8]. To develop an algorithm capable of performing real-time, multichannel frequency analysis, one needs to efficiently utilize the computer time and also improve the computational efficiency.

Since the development of the radix-2 FFT proposed by Cooley and Tukey [4], many FFT algorithms have been rapidly developed and implemented using both software and hardware approaches [2], [3], [6], [7], [10], [13]–[18], [20]. Of these, the split-radix FFT (sr-FFT) algorithm derived by Duhamel and Hollmann [6], [7] has a simpler structure and better computational efficiency. The work of Duhamel and Hollmann has been further discussed and extended in [14], [16], [18], [20].

In fact, both the conventional Cooley-Tukey FFT algorithm and the Duhamel-Hollmann sr-FFT algorithm were developed to start building a butterfly structure for the FFT computation only after all the data points had been collected. Hence, we refer to this algorithm type as the "whole-block FFT algorithm." If we try to design the real-time STFT algorithm in this manner, most of the processing time is wasted on acquiring the entire epoch of the (multichannel) EEG because of the low sampling rate (200 Hz).

The processing time could be further reduced if we simultaneously build some local butterfly structures (to be called the "butterfly modules" or simply the "modules") while the data acquisition (I/O process) is still under way. These local butterfly modules perform the decomposed smaller-size FFT computation. The goal is to finish building as many butterfly modules as possible with the data points available. The authors recently presented the idea of how to implement a real-time FFT algorithm, based on the sr-FFT butterfly structure, for a complex data array [9]. In this paper, we explore the symmetrical property of the discrete Fourier transform (DFT), given that the input data are real, to further reduce the number of complex arithmetic operations in the real-time FFT algorithm. We also present our experimental results of analyzing the time span of constructing the local butterfly modules on a sample-to- sample scale. The performance of the algorithm is measured by the degree of synchronization of the data acquisition and the Fourier analysis.

2. DIT sr-FFT for real input data

A local butterfly module in the DIT sr-FFT structure looks like a horizontally reversed "L." The array size N is an integer power of 2, $N = 2^r$, where the positive integer r indicates the number of decomposed stages in the DIT sr-FFT butterfly structure. Consider a local butterfly module at the $(r-j)$th stage. As illustrated in Figure 1, the module performing an L-point DFT $G[k]$, where $L = 2^{r-j}$, involves one $\frac{L}{2}$-point DFT block, two $\frac{L}{4}$-point DFT blocks, and one $\frac{L}{2}$-point non-DFT block. The L-point DFT $G[k]$ of data array $g[n]$ is decomposed into one $\frac{L}{2}$-point

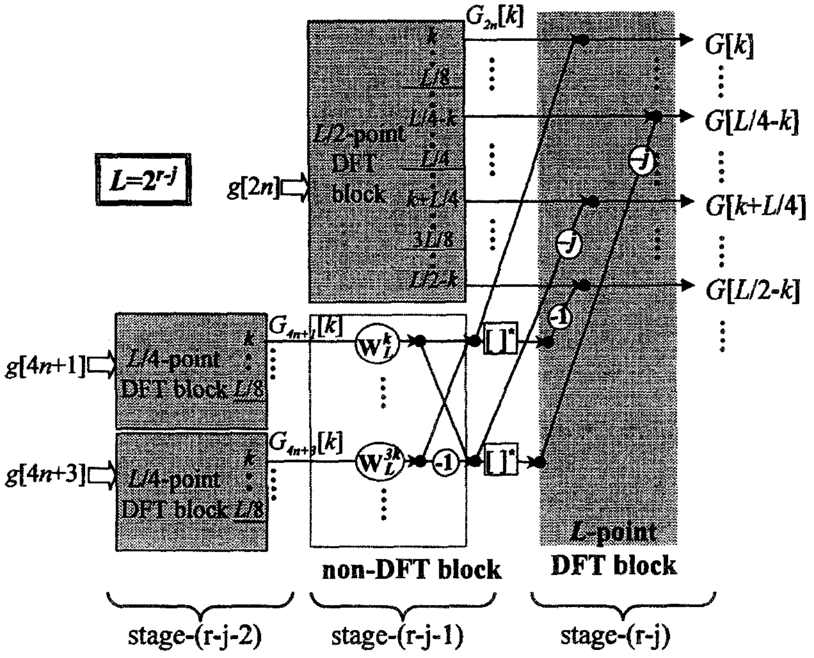

Figure 1. A local L-shaped butterfly module performing an L-point sr-FFT at the $(r - j)$th stage.

DFT $G_{2n}[k] = \text{DFT}\{g[2n]\}$ and two $\frac{L}{4}$-point DFTs, $G_{4n+1}[k] = \text{DFT}\{g[4n+1]\}$ and $G_{4n+3}[k] = \text{DFT}\{g[4n+3]\}$, with the twiddle factors

$$G[k] = \sum_{n=0}^{L-1} g[n]W_L^{nk} = \sum_{m=0}^{\frac{L}{2}-1} g[2m]W_L^{2mk} + \sum_{m=0}^{\frac{L}{2}-1} g[2m+1]W_L^{(2m+1)k}$$

$$= \sum_{m=0}^{\frac{L}{2}-1} g[2m]W_L^{2mk} + \sum_{m=0}^{\frac{L}{4}-1} g[4m+1]W_L^{(4m+1)k} + \sum_{m=0}^{\frac{L}{4}-1} g[4m+3]W_L^{(4m+3)k}, \tag{2.1}$$

where $W_L = e^{-j\frac{2\pi}{L}}$ and $0 \le k \le (L-1)$ [12]. The first term is the $\frac{L}{2}$-point DFT of the even-indexed data points in the $g[n]$ array:

$$G_{2n}[k] = \sum_{n=0}^{\frac{L}{2}-1} g[2n]W_{\frac{L}{2}}^{nk}, \tag{2.2}$$

where $0 \le k \le \left(\frac{L}{2} - 1\right)$. The last two terms in (2.1) represent the $\frac{L}{4}$-point DFTs, $G_{4n+1}[k]$ and $G_{4n+3}[k]$, with the twiddle factors

$$\sum_{n=0}^{\frac{L}{4}-1} g[4n+1]W_L^{(4n+1)k} = W_L^k \sum_{n=0}^{\frac{L}{4}-1} g[4n+1]W_{\frac{L}{4}}^{nk}$$
$$= W_L^k \cdot G_{4n+1}[k], \quad \text{and} \tag{2.3}$$
$$\sum_{n=0}^{\frac{L}{4}-1} g[4n+3]W_L^{(4n+3)k} = W_L^{3k} \sum_{n=0}^{\frac{L}{4}-1} g[4n+3]W_{\frac{L}{4}}^{nk}$$
$$= W_L^{3k} \cdot G_{4n+3}[k]\,, \tag{2.4}$$

where $0 \le k \le \left(\frac{L}{4}-1\right)$. Hence, the algorithm is a mixture of the radix-2 and radix-4 DIT FFTs.

In most applications, the input data are real. For a real sequence $x[n]$, the N-piont DFT $X[k]$ possesses the complex-conjugated symmetrical property

$$X[(-k))_N] = X[N-k] = X^*[k], \quad 1 \le k \le (N-1)\,, \tag{2.5}$$

where $((-k))_N$ denotes the circular operation on the discrete frequency variable k [12]. The symmetrical property, which halves the number of arithmetic operations, can be applied to each decomposed local DFT block to reduce the complex arithmetic operations. Thus, to compute the L-piont DFT, (2.1) can be reduced in two ways. First, consider the $\frac{L}{4}$-point DFTs multiplied by the twiddle factors in (2.3) and (2.4). If the multiplications $W_L^k \cdot G_{4n+1}[k]$ and $W_L^{3k} \cdot G_{4n+3}[k]$ for $1 \le k \le \left(\frac{L}{8}\right)$ are computed, it follows that $W_L^{k'} \cdot G_{4n+1}[k']$ and $W_L^{3k'} \cdot G_{4n+3}[k']$ for $\left(\frac{L}{8}+1\right) \le k' \le \left(\frac{L}{4}-1\right)$ can be obtained by applying the symmetrical property. That is,

$$W_L^{k'} \cdot G_{4n+1}[k'] = W_L^{\frac{L}{4}-k} \cdot G_{4n+1}\left[\frac{L}{4}-k\right]$$
$$= W_L^{\frac{L}{4}} W_L^{-k} \cdot G_{4n+1}^*[k] = -j\left(W_L^k \cdot G_{4n+1}[k]\right)^*\,, \tag{2.6}$$
$$W_L^{3k'} \cdot G_{4n+3}[k'] = W_L^{3\left(\frac{L}{4}-k\right)} \cdot G_{4n+3}\left[\frac{L}{4}-k\right]$$
$$= W_L^{\frac{3L}{4}} W_L^{-3k} \cdot G_{4n+3}^*[k] = j\left(W_L^{3k} \cdot G_{4n+3}[k]\right)^*\,, \tag{2.7}$$

where $1 \le k \le \left(\frac{L}{8}-1\right)$ and $k' = \frac{L}{4}-k$. Note that $G_{2n}[k]$ in (2.2) is defined for $0 \le k \le \left(\frac{L}{2}-1\right)$, and $G_{4n+1}[k]$ and $G_{4n+3}[k]$ are defined for $0 \le k \le \left(\frac{L}{4}-1\right)$. The complete $G[k]$ can be obtained by using the periodic property. Thus, by implementing the symmetrical property in (2.6) and (2.7), equation (2.1) for computing the first $\frac{L}{2}$-point DFT values becomes

$G[1]$ through $G\left[\frac{L}{8}\right]$:

$$G[k] = G_{2n}[k] + \left(W_L^k \cdot G_{4n+1}[k] + W_L^{3k} \cdot G_{4n+3}[k]\right), \quad 1 \le k \le \left(\frac{L}{8}\right), \tag{2.8}$$

$G\left[\frac{L}{8}+1\right]$ through $G\left[\frac{L}{4}\right]$:

$$\begin{aligned}G\left[\frac{L}{4}-k\right] &= G_{2n}\left[\frac{L}{4}-k\right] \\ &\quad + \left(W_L^{\frac{L}{4}-k}\cdot G_{4n+1}\left[\frac{L}{4}-k\right] + W_L^{3\left(\frac{L}{4}-k\right)}\cdot G_{4n+3}\left[\frac{L}{4}-k\right]\right), \\ &= G_{2n}\left[\frac{L}{4}-k\right] - j\left(W_L^k\cdot G_{4n+1}[k] - W_L^{3k}\cdot G_{4n+3}[k]\right)^*, \\ &\qquad 0\le k\le\left(\frac{L}{8}-1\right), \qquad (2.9)\end{aligned}$$

$G\left[\frac{L}{4}+1\right]$ through $G\left[\frac{3L}{8}\right]$:

$$\begin{aligned}G\left[k+\frac{L}{4}\right] &= G_{2n}\left[k+\frac{L}{4}\right] - j\left(W_L^k\cdot G_{4n+1}[k] - W_L^{3k}\cdot G_{4n+3}[k]\right), \\ &\qquad 1\le k\le\left(\frac{L}{8}\right), \qquad (2.10)\end{aligned}$$

$G\left[\frac{3L}{8}+1\right]$ through $G\left[\frac{L}{2}-1\right]$:

$$\begin{aligned}G\left[\frac{L}{2}-k\right] &= G_{2n}\left[\frac{L}{2}-k\right] + \left(W_L^{\frac{L}{2}-k}\cdot G_{4n+1}\left[\frac{L}{2}-k-\frac{L}{4}\right]\right. \\ &\qquad \left. + W_L^{3\left(\frac{L}{2}-k\right)}\cdot G_{4n+3}\left[\frac{L}{2}-k-\frac{L}{4}\right]\right) \\ &= G_{2n}\left[\frac{L}{2}-k\right] - \left(W_L^k\cdot G_{4n+1}[k] + W_L^{3k}\cdot G_{4n+3}[k]\right)^*, \\ &\qquad 1\le k\le\left(\frac{L}{8}-1\right). \qquad (2.11)\end{aligned}$$

Notice that computing $G[0]$ and $G\left[\frac{L}{2}\right]$ involves totally three real additions:

$$G[0] = G_{2n}[0] + (G_{4n+1}[0] + G_{4n+3}[0]) \qquad (2.12)$$

$$G\left[\frac{L}{2}\right] = G_{2n}[0] - (G_{4n+1}[0] + G_{4n+3}[0]) \; . \qquad (2.13)$$

The remaining half of the DFT values $G[k]$ for $\left(\frac{L}{2}+1\right)\le k\le(L-1)$ can be obtained from the complex conjugation of $G[k]$, $1\le k\le\left(\frac{L}{2}-1\right)$:

$G\left[\frac{L}{2}+1\right]$ through $G\left[\frac{3L}{4}-1\right]$:

$$G\left[k+\frac{L}{2}\right]=G^*\left[L-\left(k+\frac{L}{2}\right)\right]$$
$$=G^*\left[\frac{L}{2}-k\right],\quad 1\le k\le\left(\frac{L}{4}-1\right),\tag{2.14}$$

$G\left[\frac{3L}{4}\right]$ through $G[L-1]$:

$$G\left[k+\frac{3L}{4}\right]=G^*\left[L-\left(k+\frac{3L}{4}\right)\right]$$
$$=G^*\left[\frac{L}{4}-k\right],\quad 0\le k\le\left(\frac{L}{4}-1\right),\tag{2.15}$$

according to the complex-conjugated symmetrical property.

Figure 1 shows the local butterfly module following the computations of equations (2.8) through (2.13). The four output sets compute the first half of the DFT samples $G[0]$ through $G\left[\frac{L}{2}\right]$, from which the rest of the DFT samples $G\left[\frac{L}{2}+1\right]$ through $G[L-1]$ can be derived using (2.14) and (2.15). The shaded regions of the Figure represent the DFT blocks: that is, the connecting branches in the regions are able to accomplish the local DFT computation. The blank region at the $(r-j-1)$th stage represents the non-DFT block.

3. Real-time implementation of DIT sr-FFT

3.1. Construction of the on-line butterfly modules.

The conventional FFT, designed to obtain a complete frequency-band spectrum, makes an inherent assumption that the input and output sequences have the same array size. However, two situations often encountered are: (1) only a narrow frequency band of the Fourier spectrum is of interest, and (2) the input data array consists of many zero-valued data points due to zero- padding. To reduce unnecessary arithmetic operations in these situations, a number of "pruned FFT" algorithms have been introduced [1], [10], [11], [13], [14], [17], [19]. Pruning is a modification of the complete butterfly structure. Input pruning removes those branches connected to the zero-valued input; output pruning removes those connected to the unattended output samples. Inspired by the input pruning scheme, the authors developed the real-time FFT algorithm, which builds the on-line butterfly modules based on the DIT sr-FFT structure.

When performing the on-line Fourier analysis of the multichannel EEG, one can take advantage of the low sampling rate and efficiently utilize the computer time. The EEG is usually sampled at 200 Hz. To analyze and display the on-line, running Fourier spectra using a 512-piont DFT, one would spend more than 2.5 seconds on acquiring the digital EEG data. The efficient algorithm presented in this paper utilizes the data points available to construct the local butterfly modules while the input data array is still not completely filled up. In Section 4 we will show that the real-time implementation of the FFT algorithm provides better synchronization capability than the conventional whole-block implementation. The more substantial improvement is observed especially for a larger number of recording channels and a larger FFT size.

To better illustrate how the algorithm works, the complete 16-piont DIT sr-FFT structure is considered, without implementing the symmetrical property. Figure 2 displays the complete butterfly structure for $X[k] = \text{DFT}\{x[n]\}$. Consider the four blocks at the second stage. The shaded regions indicate the DFT blocks; the blank region represents the non-DFT block. Further decomposition in the non-DFT block at the second stage results in two length-2 DFTs at the first stage, directly connected to the input data array.

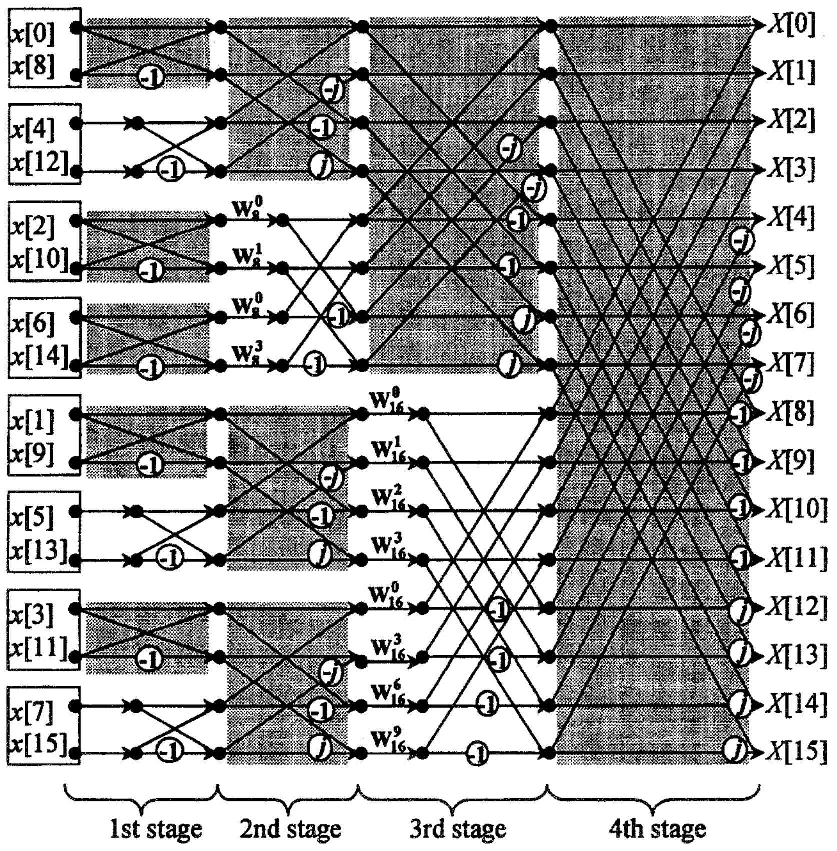

Figure 2. The signal flow graph of a 16-point DIT sr-FFT structure.

The signal flow graphs in Figure 3 are used to illustrate the progress of the 16-point sr-FFT butterfly structure when the data points collected are $x[i]$, $i = 0, \ldots, 12$ (Figure 3a) and $0, \ldots, 14$ (Figure 3 b). According to the DIT FFT algorithm, the input sequence is arranged in bit-reversed order to obtain a normal-order output sequence for in-place computation. Hence, in Figure 3 the input data array is arranged, from the top downwards, in bit-reversed order at the input stage of the DIT sr-FFT structure. Note that the array index i in Figure 3 is expressed in normal bit order, that is, the actual order of the collected input sequence. The solid lines represent the accomplished computational branches. The dashed lines indicate that the signal-flow branches are not constructed because the last few data points have not been collected. Obviously, the first butterfly module

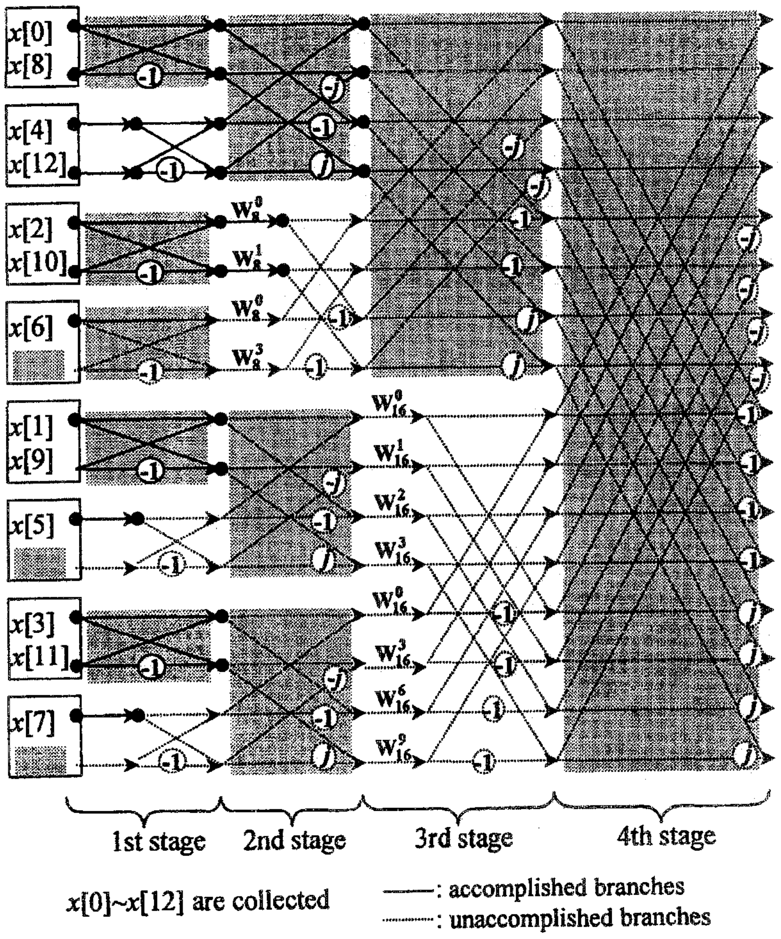

Figure 3a. The signal flow graphs showing the accomplished modules and computational branches upon receipt of $x[0]$ through $x[12]$.

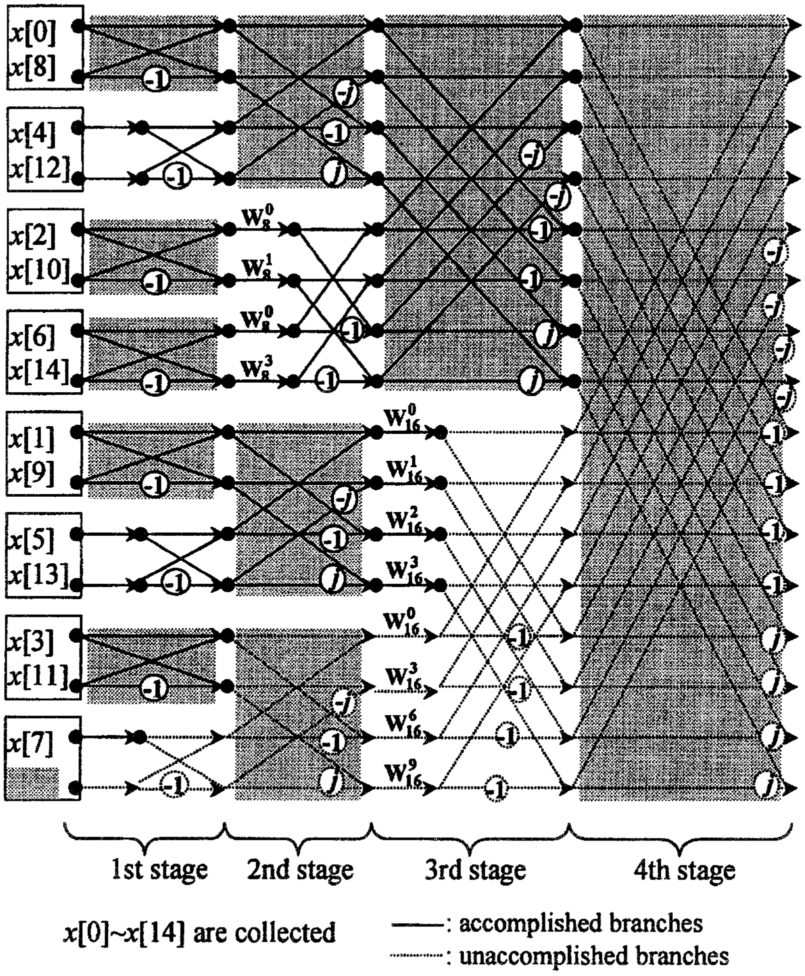

Figure 3b. The signal flow graphs showing the accomplished modules and computational branches upon receipt of $x[0]$ through $x[14]$, for an FFT size of 16.

is constructed when the $\left(\frac{N}{2} + \left(\frac{N}{4} + 1\right)\right)$th point ($x[12]$ in Figure 3a) is available, or when the construction of signal-flow branches proceeds to the second stage be cause the 4-point DFT is the basic (smallest-sized) butterfly module according to the DIT sr-FFT structure. Notice that the last block at each stage cannot be accomplished until the last point $x[N-1]$ is collected (Figure 3b). Then one question arises: Given that the nth $\left(\frac{N}{2} + \frac{N}{4} + 1 \leq n \leq N-1\right)$ position of the input data array is filled, which butterfly modules can be built? In the following section the authors propose a strategy of identifying these butterfly modules and directing the growth of signal-flow branches.

3.2. Sequence of constructing local butterfly modules.

Consider an N-point DFT where $N = 2^r$. The indexes of input data array are expressed in binary form (r-bit): $x[0\ldots 0]$ through $x[1\ldots 1]$. The array indexes that we refer to in this section are all designated in the normal order. Let the first bit represent the least significant bit. There are r stages in the DIT sr-FFT butterfly structure, including the first stage, at which no butterfly module is constructed.

Figure 4 displays the status of the last four stages. The shaded regions represent the DFT blocks. The blank regions are the non-DFT blocks, which require further decomposition into two half-length DFTs. At any given stage, a DFT block is constructed from the results of one DFT and one non-DFT block at the previous stage, whereas a non-DFT block is constructed from the results of two DFT blocks at the previous stage. Figure 4 shows that, to complete the first block (a DFT block) at the ith stage, one must have at least collected the input data point $x[n]$, where $n \geq 2^{r-1} + 2^{r-2} + \cdots + 2^{r-i}$. That is, the last (higher-significant) i bits of the array index must be 1. For example, the top DFT block of the second stage of a 16-point sr-FFT can be completed when $x[1100]$ is collected.

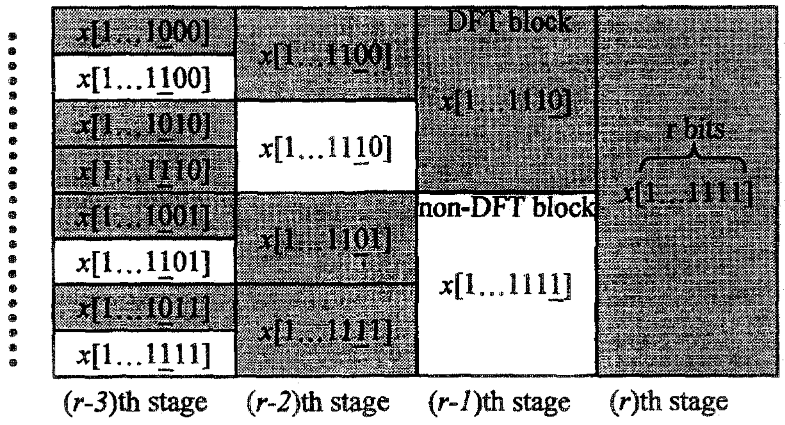

Figure 4. The DFT (shaded) and non-DFT (blank) blocks partitioning the last four stages of an r-stage DIT sr-FFT butterfly structure.

The authors also observed a straightforward regulation for the real-time FFT algorithm to identify the butterfly modules ready to be constructed. Consider the array index n_b (in binary format) at the $(r - j)$th stage. If the jth bit of n_b is 0, one cannot move to the $(r - j + 1)$th stage. On the other hand, construction of the butterfly module can further proceed to the next higher stage if the jth bits is 1 at the $(r - j)$th stage. As indicated in Figure 4, we can accomplish the top DFT block at the $(r - 1)$th stage after the second block (non-DFT) at the $(r - 2)$th stage is accomplished upon receipt of $x[1\ldots 11\underline{1}0]$. In other words, the appearance of $x[1\ldots 11\underline{1}0]$ enables the algorithm to construct the (2^{r-2})th block (from the top) of the first stage, the (2^{r-3})th block of the second stage, and so on, continuing up to the (2^0)th block (the first block) of the $(r-1)$th stage. Alternatively speaking, the number of consecutive bit 1's, $n$bit1, counted from the most significant bit position (the leftmost bit position) determines the task to be performed upon receipt of $x[n_b]$. If $n$bit1 is an even number, the algorithm constructs the butterfly modules as follows: non-DFT (first stage) $\rightarrow$ DFT (second stage) $\rightarrow \cdots \rightarrow$ non-DFT (($n$bit1 $-$ 1)th stage) $\rightarrow$ DFT (($n$bit1)th stage). If $n$bit1 is an odd number, the algorithm performs the module construction: DFT (first stage) $\rightarrow$ non-DFT (second stage) $\rightarrow \cdots \rightarrow$ non-DFT (($n$bit1 $-$ 1)th stage) $\rightarrow$ DFT (($n$bit1)th stage). That is, the first stage performs the non-DFT (DFT) block construction if $n$bit1 is an even (odd) number. Consequently, the guideline directs the real-time FFT algorithm to correctly construct the succeeding butterfly modules.

Next we evaluate the computational complexity of the real-time FFT algorithm and compare the synchronous capability of both the real-time and whole-block FFT algorithms.

4. Comparison of efficiency of real-time and whole-block implementations of FFT

The computational complexity of the sr-FFT algorithm has been discussed in detail in [6], [7], [9], [17]. The authors previously developed an alternative method of analyzing the number of complex multiplications and additions, for a complex input data array, based on the number of DFT blocks at each stage [9]. When adapted for the real input data array, equations (2.8) through (2.15) can be applied.

Consider an $N(=2^r)$-point DIT sr-FFT of a real input data array. As a result of implementing (2.8) through (2.15), the number of complex additions in an L-point DFT block (Figure 1) is $\frac{3L}{4}$ instead of $\frac{6L}{4}$ without implementing the symmetrical property [9]. Three real additions are required to compute $G[0]$ and $G\left[\frac{L}{2}\right]$. From the derivation in the Appendix, the total number of complex additions $N_{\text{total}}(\text{ADD}_\text{c})$ is

$$N_{\text{total}}(\text{ADD}_\text{c}) = \left(\frac{r}{2} - \frac{2}{3}\right)N - \frac{1}{3}(-1)^r\,, \quad r > 2 \tag{4.1}$$

and the total number of real additions $N_{\text{total}}(\text{ADD}_\text{r})$ is

$$N_{\text{total}}(\text{ADD}_\text{r}) = \begin{cases} \frac{11}{6}N - \frac{5}{3}, & \text{for } r \text{ odd} \\ \frac{11}{6}N - \frac{4}{3}, & \text{for } r \text{ even} \end{cases} . \tag{4.2}$$

To complete a local L-point DFT block, the number of complex multiplications is $\frac{L}{4}$. The total number of complex multiplications $N_{\text{total}}(\text{MUL}_\text{c})$ is

$$N_{\text{total}}(\text{MUL}_\text{c}) = \left(\frac{r}{6} - \frac{5}{18}\right)N - \frac{2}{9}(-1)^r\,. \tag{4.3}$$

To demonstrate the synchronization ability of the real-time FFT algorithm, we calculate the number of arithmetic operations left to be completed upon receipt of the last point $x[2^r - 1]$. Because the EEG is sampled at a low rate ($200 Hz$), many computations can be performed before collecting the last data point. By real-time implementation of the FFT algorithm, only the last block of each stage is not constructed before $x[2^r - 1]$ is collected.

Note that if r is an odd number, the last block of the first stage is a DFT block; otherwise, it is a non-DFT block. In addition, the last block is a DFT block at every other stage. Let r be an odd number. The number of complex multiplications involved in completing the last block of each stage is: 0 (the first stage), 2^1 (the third stage), 2^3 (the fifth stage), and so on, continuing to 2^{r-2} (the rth stage). Thus, the total n'ımber of complex multiplications $N_{\text{last}}(\text{MUL}_\text{c})$ is

$$N_{\text{last}}(\text{MUL}_\text{c}) = 0 + 2^1 + 2^3 + \cdots + 2^{r-2} = \frac{N}{3} - \frac{2}{3}, \qquad \text{for } r \text{ odd.} \tag{4.4}$$

If r is an even number, the number of complex multiplications is: 2^2 (the fourth stage), and so on, continuing to 2^{r-2} (the rth stage). The total number becomes

$$N_{\text{last}}(\text{MUL}_\text{c}) = 2^2 + 2^4 + \cdots + 2^{r-2} = \frac{N}{3} - \frac{4}{3}, \qquad \text{for } r \text{ even.} \tag{4.5}$$

Similarly, the number of complex additions involved in completing the last block of each stage is: $3 \cdot 2^1$ (the third stage), $3 \cdot 2^3$ (the fifth stage) and so on, continuing to $3 \cdot 2^{r-2}$ (the rth stage) for an odd-numbered r. Thus, the total number of complex additions $N_{\text{last}}(\text{ADD}_\text{c})$ is

$$N_{\text{last}}(\text{ADD}_\text{c}) = 3 \cdot 2 + 3 \cdot 2^3 + \cdots + 3 \cdot 2^{r-2} = N - 2, \qquad \text{for } r \text{ odd.} \tag{4.6}$$

For an even-numbered r, the number of complex additions is: 1 (the second stage), $3 \cdot 2^2$ (the fourth stage), and so on, continuing to $3 \cdot 2^{r-2}$ (the rth stage). The total number becomes

$$N_{\text{last}}(\text{ADD}_\text{c}) = 1 + 3 \cdot 2^2 + \cdots + 3 \cdot 2^{r-2} = N - 3, \qquad \text{for } r \text{ even.} \tag{4.7}$$

And the number of real additions $N_{\text{last}}(\text{ADD}_\text{r})$ is

$$N_{\text{last}}(\text{ADD}_\text{r}) = \begin{cases} \frac{3r}{2} - 1, & \text{for } r \text{ even} \\ \frac{3r}{2} + \frac{1}{2}, & \text{for } r \text{ odd} \end{cases} . \tag{4.8}$$

Equations (4.4) through (4.7), compared with (4.1) and (4.3), show that both the ratios of $N_{\text{last}}(\text{MUL}_\text{c})$ to $N_{\text{total}}(\text{MUL}_\text{c})$ and of $N_{\text{last}}(\text{ADD}_\text{c})$ to $N_{\text{total}}(\text{ADD}_\text{c})$ are approximately proportionate to $\frac{2}{r}$. That is, after acquiring all the data points, the extra computational time required by the real-time FFT algoirthm is only $\frac{2}{r}$ the time required by the conventional whole-bloc algorithm. Hence, the real-time FFT algorithm offers better synchronization capability, especially for a large N (the FFT size) when applied to on-line Fourier analysis.

For comparison, the authors experimented with the DIT sr-FFT algorithm implemented in the real-time and whole-block mode. Consider a 512-point DFT. The real-time sr-FFT algorithm was run on a Pentium 66 and on a 486 DX-100. Table

1 lists the average times (the second and the third columns) spent building the local butterfly modules when $x[i]$, $0 \leq i \leq 511$, is collected. The data points are grouped according to the computational time (number of arithmetic operations) required for construction of the local modules. As discussed in Section 3.1, the real-time FFT algorithm starts building the first module when $x[384]$ ($x\left[\frac{N}{2}+\frac{N}{4}\right]$, or $x[110000000]$) is collected. Until it collects the $x[448]$ ($x[111000000]$), the real-time FFT algorithm cannot proceed to the third stage. The average time spent on constructing the 4-point butterfly module (group II) is 3.86 μsec on a Pentium 66 and 5.72 μsec on a 486 DX-100. While acquiring the data points from $x[448]$ to $x[479]$ (group III), the algorithm is constructing the 8-point butterfly modules at the third stage, which takes 5.26 μsec (Pentium 66) and 9.62 μsec (486 DX-100) on an average. The process of constructing the total butterfly modules continues with collection of each data point. The total time, which is obtained by summing up the 512 values of average time in the second and the third columns, for the real-time FFT algorithm to perform the 512-point DFT is 2.06 msec on a Pentinum 66 and 3.83 msec on a 486 DX-100. In the on-line applications, the speed of data acquisition (i.e., the sampling rate) dominates the entire processing time. Consider the sampling rate of 200 Hz frequently employed in the EEG data acquisition. It takes 2.56 sec to acquire the 512 data points. However, upon receipt of the last data point $x[511]$, the real-time algorithm only takes 0.43 msec (Pentium 66) to complete the rest of the butterfly modules. The conventional whole-block FFT algorithm takes approximately 2.01 msec on a Pentium 66 and 3.88 msec on a 486 DX-100, in addition to the 2.56 sec required for data acquisition.

Table 1. Computer time spent building local butterfly modules upon receipt of $x[i]$ for FFT size of 512

Group of data points collected	Average time (μsec) (Pentium 66)	Average time (μsec) (486 DX-100)	Stage in progress	Size of local module
I : $x[0] \sim x[383]$	0.57	0.62	1st	2-point
II : $x[384] \sim x[447]$	3.86	5.72	2nd	4-point
III : $x[448] \sim x[479]$	5.26	9.62	3rd	8-point
IV : $x[480] \sim x[495]$	12.33	23.44	4th	16-point
V : $x[496] \sim x[503]$	23.64	47.93	5th	32-point
VI : $x[504] \sim x[507]$	50.19	103.30	6th	64-point
VII : $x[508] \sim x[509]$	100.46	210.51	7th	128-point
VIII : $x[510]$	206.57	433.58	8th	256-point
VIIII : $x[511]$	432.71	894.68	9th	512-point

5. Conclusion

This paper presents a new method of implementing the DIT sr-FFT algorithm in the real-time mode. The real-time FFT algorithm simultaneously constructs the local butterfly modules while the data acquisition (signal recording, digitization, and collection) process is under way. Compared with the conventional whole-block sr-FFT algorithm, the real-time FFT algorithm synchronizes better with the real process, especially for the multichannel implementation. The results shown in Section 4 support the basis of this study. Consider the example of performing the on-line Fourier analysis (FFT size of 512) of the 64-channel EEG. In addition to the 2.56 sec required for data acquisition, the whole-block FFT algoirthm causes a delay of (2.01 msec) $\times$ 64 = 128.64 msec on a Pentium 66. This delay is reduced to (0.43 msec) $\times$ 64 = 27.69 msec when using the real-time implementation of the FFT algorithm.

Notice that, upon receipt of the last data point, the computational time spent on building the rest of the local butterfly modules is approximately 0.43 msec, that is, 21% of the total time needed to complete the 512-point sr-FFT. The result is consistent with our derivation in Section 4, i.e., $N_{\text{last}}/N_{\text{total}} \approx 2/r$ (here, $r = 9$). This fact shows that increasing the FFT size causes the synchronization capability of the real-time FFT algorithm over the conventional whole-block FFT algorithm to become more evident.

Appendix

Given that the FFT size $N = 2^r$, there are 2^j blocks at the $(r-j)$th stage (Figure 4). Note that the non-DFT blocks at the $(r-j)$th stage are decomposed from the DFT blocks at the $(r-j+1)$th stage. Hence, the number of DFT blocks at the $(r-j)$th stage is

$$N_{r-j}(\text{DFT}) = 2^j - N_{r-j+1}(\text{DFT}), \quad j = 1, \ldots, r-1, \tag{A.1}$$

and $N_r(\text{DFT}) = 2^0 = 1$ for the last stage. It follows that

$$\begin{aligned} N_{r-j}(\text{DFT}) &= \sum_{l=0}^{j} (-1)^l 2^{j-l} \\ &= 2^j \cdot \left[\frac{2}{3} + \frac{1}{3}\left(-\frac{1}{2}\right)^j\right], \quad j = 1, \ldots, r-1. \end{aligned} \tag{A.2}$$

Therefore,

$$N_1(\text{DFT}) = \frac{N}{3} + \frac{1}{3}(-1)^{r-1}, \tag{A.3a}$$

$$N_2(\text{DFT}) = \frac{N}{3 \cdot 2} + \frac{1}{3}(-1)^{r-2}, \tag{A.3b}$$

$$N_3(\text{DFT}) = \frac{N}{3 \cdot 2^2} + \frac{1}{3}(-1)^{r-3}\,, \tag{A.3.c}$$

$$\vdots$$

$$N_{r-1}(\text{DFT}) = \frac{N}{3 \cdot 2^{r-2}} + \frac{1}{3}(-1)^1 = 1\,. \tag{A.3d}$$

At the first stage of the DIT sr-FFT structure, the algorithm requires $2 \cdot 2^{r-1} = N$ real additions and no complex arithmetic operation. At the second stage, computing the 4-point DFT block requires two real additions and one complex addition. According to (A.3b), the algorithm performs

$\frac{N}{3} + \frac{2}{3}(-1)^{r-2}$ real additions,
$\frac{N}{6} + \frac{1}{3}(-1)^{r-2}$ complex additions,

and no multiplications.

To calculate the number of arithmetic operations at the $(r-j)$th stage $(0 \le j \le r-3)$, we examine the butterfly module in Figure 1. Note that, to accomplish the L-point DFT $(L \ge 8)$ at the $(r-j)$th stage, the structural schema consists of one $\frac{L}{2}$-point DFT block and one non-DFT block at the $(r-j-1)$th stage as well as two $\frac{L}{4}$-point DFT blocks at the $(r-j-2)$th stage, where $L = 2^{r-j} = N/2^j$.

Suppose that $G_{2n}[k]$, $G_{4n+1}[k]$, and $G_{4n+3}[k]$ have been computed. According to equations (2.8) through (2.15), computing the L-point DFT $G[k]$ from $G_{2n}[k]$, $G_{4n+1}[k]$, and $G_{4n+3}[k]$ requires $\frac{L}{4} = 2^{r-j-2}$ complex multiplications (because of the twiddle factors in the non-DFT block) and $\frac{3L}{4} = 3 \cdot 2^{r-j-2}$ complex additions as well as three real additions for computing $G[0]$ and $G\left[\frac{L}{2}\right]$. These number already include the arithmetic operations in the non-DFT block at the $(r-j-1)$th stage. Then, at the $(r-j)$th stage, the total numbers of complex multiplications $N_{r-j}(\text{MUL}_\text{c})$, complex additions $N_{r-j}(\text{ADD}_\text{c})$, and real additions $N_{r-j}(\text{ADD}_\text{r})$ are calculated as follows:

$$\begin{aligned} N_{r-j}\,(\text{MUL}_\text{c}) &= 2^{r-j-2} \cdot N_{r-j}(\text{DFT}) \\ &= 2^{r-2} \cdot \left[\frac{2}{3} + \frac{1}{3}\left(-\frac{1}{2}\right)^j\right], \quad 0 \le j \le r-3 \end{aligned} \tag{A.4}$$

$$\begin{aligned} N_{r-j}\,(\text{ADD}_\text{c}) &= \left(3 \cdot 2^{r-j-2}\right) \cdot N_{r-j}(\text{DFT}) \\ &= \left(3 \cdot 2^{r-2}\right) \cdot \left[\frac{2}{3} + \frac{1}{3}\left(-\frac{1}{2}\right)^j\right], \quad 0 \le j \le r-3 \end{aligned} \tag{A.5}$$

$$\begin{aligned} N_{r-j}\,(\text{ADD}_\text{r}) &= 3 \cdot N_{r-j}(\text{DFT}) = 2^j \cdot \left[2 + \left(-\frac{1}{2}\right)^j\right] \\ &= 2^{j+1} + (-1)^j, \quad 0 \le j \le r-3\,. \end{aligned} \tag{A.6}$$

Finally, the total number of complex multiplications $N_{\text{total}}(\text{MUL}_c)$ is

$$\begin{aligned} N_{\text{total}}(\text{MUL}_c) &= \sum_{j=0}^{r-3} N_{r-j}(\text{MUL}_c) = 2^{r-2} \cdot \sum_{j=0}^{r-3}\left[\frac{2}{3}+\frac{1}{3}\left(-\frac{1}{2}\right)^j\right] \\ &= \frac{1}{3}(r-2)2^{r-1}+\frac{1}{9}2^{r-1}-\frac{2}{9}(-1)^{r-2} \\ &= \left(\frac{r}{6}-\frac{5}{18}\right)N-\frac{2}{9}(-1)^r, \end{aligned} \tag{A.7}$$

the total number of complex additions $N_{\text{total}}(\text{ADD}_c)$ is

$$\begin{aligned} N_{\text{total}}(\text{ADD}_c) &= \frac{N}{6}+\frac{1}{3}(-1)^{r-2}+\sum_{j=0}^{r-3} N_{r-j}(\text{ADD}_c) \\ &= \frac{N}{6}+\frac{1}{3}(-1)^{r-2}+3\cdot 2^{r-2}\cdot\sum_{j=0}^{r-3}\left[\frac{2}{3}+\frac{1}{3}\left(-\frac{1}{2}\right)^j\right] \\ &= \left(\frac{r}{2}-\frac{2}{3}\right)\cdot 2^r-\frac{1}{3}(-1)^{r-2} = \left(\frac{r}{2}-\frac{2}{3}\right)N-\frac{1}{3}(-1)^r, \end{aligned} \tag{A.8}$$

and the total number of real additions $N_{\text{total}}(\text{ADD}_r)$ is

$$\begin{aligned} N_{\text{total}}(\text{ADD}_r) &= N+\frac{N}{3}+\frac{2}{3}(-1)^{r-2}+\sum_{j=0}^{r-3} N_{r-j}(\text{ADD}_r) \\ &= \frac{4N}{3}+\frac{2}{3}(-1)^{r-2}+\sum_{j=0}^{r-3}\left[2^{j+1}+(-1)^j\right] \\ &= \frac{11}{6}2^r-2+\frac{2}{3}(-1)^{r-2}+\sum_{j=0}^{r-3}(-1)^j \\ &= \begin{cases} \frac{11}{6}N-\frac{5}{3}, & \text{for } r \text{ odd} \\ \frac{11}{6}N-\frac{4}{3}, & \text{for } r \text{ even} \end{cases}. \end{aligned} \tag{A.9}$$

References

[1] S. Barash and Y. Ritov, Logarithmic pruning of FFT frequencies, *IEEE Trans. Signal Processling*, 41 (3), 1398–1400, 1993.

[2] R. G. Bickford, Computer analysis of background activity, Rémond, A. (ed.), *EEG Informatics*, Elsevier, Amsterdam, 1977, pp. 215–232.

[3] V. Boriakoff, FFT computation with systolic arrays: A new architecture, *IEEE Trans. Circuits Systems — II: Analog Digital Signal Process.*, 41(4), 278–284, 1994.

[4] J. W. Cooley and J. W. Tukey, An algorithm for machine computation of complex Fourier series, *Math. Comp.*, 19, 297–301, 1965.

[5] R. Cooper, J. W. Osselton, and J. C. Shaw, *EEG Technology*, 3rd ed., Butterworth, Woburn, MA, 1980, Chapter 6.

[6] P. Duhamel, Implementation of "split-radix" FFT algorithms for complex, real and real-symmetric data, *IEEE Trans. Acoust. Speech Signal Process.*, ASSP-34, 285–295, 1986.

[7] P. Duhamel and H. Hollmann, Split radix FFT algorithm, *Electronics Lett.*, 20(1), 14–16, 1984.

[8] A. Gevins, P. Brickett, B. Costales, J. Le, and B. Reutter, Beyond topographic mapping: Towards functional-anatomical imaging with 124-channel EEGs and 3-D MRIs, *Brain Topography*, 3, 53–64, 1990.

[9] P. C. Lo and Y. Y. Lee, Real-time implementation of the split-radix FFT — An algorithm to efficiently construct local butterfly modules, *Signal Processing*, submitted.

[10] J. D. Markel, FFT pruning, *IEEE Trans. Audio Electroacoust.*, AU-19(4), 305–311, 1971.

[11] K. Nagai, Pruning the decimation-in-time FFT algorithm with frequency shift, *IEEE Trans. Acoust. Speech Signal Process.*, ASSP-34(4), 1008–1010, 1986.

[12] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1989, Chapter 9.

[13] D. E. Panera, S. R. Mani, and S. H. Nawab, STFT computation using pruned FFT algorithms, *IEEE Signal Processing Lett.*, 1(4), 61–63, 1994.

[14] C. Roche, A split-radix partial input/output fast Fourier transform algorithm, *IEEE Trans. Signal Process.*, 40(5), 1273–1276, 1992.

[15] I. W. Selesnick and C. S. Burrus, Automatic generation of prime length FFT programs, *IEEE Trans. Signal Process.*, 44(1), 14–24, 1996.

[16] A. N. Skodras, Effecient computation of the split-radix FFT, *IEEE Proceedings — F*, 139(1), 56–60, 1992.

[17] H. V. Sorensen and C. S. Burrus, Efficient computation of the DFT with only a subset of input or output points, *IEEE Trans. Signal Process.*, 41(3), 1184–1200, 1993.

[18] H. V. Sorensen, M. T. Heideman, and C. S. Burrus, On computing the split-radix FFT, *IEEE Trans. Acoust. Speech Signal Process.*, ASSP-34(1), 152–156, 1986.

[19] T. V. Sreenivas and P. V. S. Rao, High resolution narrow-band spectra by FFT pruning, *IEEE Trans. Acoust. Speech Signal Process.*, ASSP-28(2), 254–257, 1980.

[20] P. R. Uniyal, Transforming real-valued sequences: Fast Fourier versus fast Hartley transform algorithms, *IEEE Trans. Signal Process.*, 42(11), 3249–3254, 1994.