Multidimensional Index Mapping*

C. Sidney Burrus

Abstract

A change of index variable or an index mapping is used to uncouple the calculations of the discrete Fourier transform (DFT). This can result is a significant reduction in the required arithmetic and the resulting algorithm is called the fast Fourier transform (FFT).

A powerful approach to the development of efficient algorithms is to break a large problem into multiple small ones. One method for doing this with both the DFT and convolution uses a linear change of index variables to map the original one-dimensional problem into a multi-dimensional problem. This approach provides a unified derivation of the Cooley-Tukey FFT, the prime factor algorithm (PFA) FFT, and the Winograd Fourier transform algorithm (WFTA) FFT. It can also be applied directly to convolution to break it down into multiple short convolutions that can be executed faster than a direct implementation. It is often easy to translate an algorithm using index mapping into an efficient program.

The basic definition of the discrete Fourier transform (DFT) is

$$C(k) = \sum_{n=0}^{N-1} x(n) \quad W_N^{nk} \tag{1}$$

where n, k, and N are integers, $j = \sqrt{-1}$, the basis functions are the N roots of unity,

$$W_N = e^{-j2\pi/N} \tag{2}$$

and $k = 0, 1, 2, \cdots, N-1$.

If the N values of the transform are calculated from the N values of the data, $x(n)$, it is easily seen that N^2 complex multiplications and approximately that same number of complex additions are required. One method for reducing this required arithmetic is to use an index mapping (a change of variables) to change the one-dimensional DFT into a two- or higher dimensional DFT. This is one of the ideas behind the very efficient Cooley-Tukey [6] and Winograd [19] algorithms. The purpose of index mapping is to change a large problem into several easier ones [3], [7]. This is sometimes called the "divide and conquer" approach [2] but a more accurate description would be "organize and share" which explains the process of redundancy removal or reduction.

1 The Index Map

For a length-N sequence, the time index takes on the values

$$n = 0, 1, 2, ..., N-1 \tag{3}$$

*Version 1.11: Sep 18, 2009 4:55 pm GMT-5
†http://creativecommons.org/licenses/by/2.0/

When the length of the DFT is not prime, N can be factored as $N = N_1 N_2$ and two new independent variables can be defined over the ranges

$$n_1 = 0, 1, 2, ..., N_1 - 1 \tag{4}$$

$$n_2 = 0, 1, 2, ..., N_2 - 1 \tag{5}$$

A linear change of variables is defined which maps n_1 and n_2 to n and is expressed by

$$n = ((K_1 n_1 + K_2 n_2))_N \tag{6}$$

where K_i are integers and the notation $((x))_N$ denotes the integer residue of x modulo N[13]. This map defines a relation between all possible combinations of n_1 and n_2 in (4) and (5) and the values for n in (3). The question as to whether all of the n in (3) are represented, i.e., whether the map is one-to-one (unique), has been answered in [3] showing that certain integer K_i always exist such that the map in (6) is one-to-one. Two cases must be considered.

1.1 Case 1.

N_1 and N_2 are relatively prime, i.e., the greatest common divisor $(N_1, N_2) = 1$.
The integer map of (6) is one-to-one if and only if:

$$(K_1 = aN_2) \quad \text{and/or} \quad (K_2 = bN_1) \quad \text{and} \quad (K_1, N_1) = (K_2, N_2) = 1 \tag{7}$$

where a and b are integers.

1.2 Case 2.

N_1 and N_2 are not relatively prime, i.e., $(N_1, N_2) > 1$.
The integer map of (6) is one-to-one if and only if:

$$(K_1 = aN_2) \quad \text{and} \quad (K_2 \neq bN_1) \quad \text{and} \quad (a, N_1) = (K_2, N_2) = 1 \tag{8}$$

or

$$(K_1 \neq aN_2) \quad \text{and} \quad (K_2 = bN_1) \quad \text{and} \quad (K_1, N_1) = (b, N_2) = 1 \tag{9}$$

Reference [3] should be consulted for the details of these conditions and examples. Two classes of index maps are defined from these conditions.

1.3 Type-One Index Map:

The map of (6) is called a type-one map when integers a and b exist such that

$$K_1 = aN_2 \quad \text{and} \quad K_2 = bN_1 \tag{10}$$

1.4 Type-Two Index Map:

The map of (6) is called a type-two map when when integers a and b exist such that

$$K_1 = aN_2 \quad \text{or} \quad K_2 = bN_1, \quad \text{but not both.} \tag{11}$$

The type-one can be used **only** if the factors of N are relatively prime, but the type-two can be used whether they are relatively prime or not. Good [9], Thomas, and Winograd [19] all used the type-one map in their DFT algorithms. Cooley and Tukey [6] used the type-two in their algorithms, both for a fixed radix $\left(N = R^M\right)$ and a mixed radix [16].

The frequency index is defined by a map similar to (6) as

$$k = \left(\left(K_3 k_1 + K_4 k_2\right)\right)_N \tag{12}$$

where the same conditions, (7) and (8), are used for determining the uniqueness of this map in terms of the integers K_3 and K_4.

Two-dimensional arrays for the input data and its DFT are defined using these index maps to give

$$\hat{x}\left(n_1, n_2\right) = x\left(\left(K_1 n_1 + K_2 n_2\right)\right)_N \tag{13}$$

$$\hat{X}\left(k_1, k_2\right) = X\left(\left(K_3 k_1 + K_4 k_2\right)\right)_N \tag{14}$$

In some of the following equations, the residue reduction notation will be omitted for clarity. These changes of variables applied to the definition of the DFT given in (1) give

$$C\left(k\right) = \sum_{n_2=0}^{N_2-1} \sum_{n_1=0}^{N_1-1} x\left(n\right) \; W_N^{K_1 K_3 n_1 k_1} \; W_N^{K_1 K_4 n_1 k_2} \; W_N^{K_2 K_3 n_2 k_1} \; W_N^{K_2 K_4 n_2 k_2} \tag{15}$$

where all of the exponents are evaluated modulo N.

The amount of arithmetic required to calculate (15) is the same as in the direct calculation of (1). However, because of the special nature of the DFT, the integer constants K_i can be chosen in such a way that the calculations are "uncoupled" and the arithmetic is reduced. The requirements for this are

$$\left(\left(K_1 K_4\right)\right)_N = 0 \quad \text{and/or} \quad \left(\left(K_2 K_3\right)\right)_N = 0 \tag{16}$$

When this condition and those for uniqueness in (6) are applied, it is found that the K_i may **always** be chosen such that one of the terms in (16) is zero. If the N_i are relatively prime, it is always possible to make **both** terms zero. If the N_i are not relatively prime, only one of the terms can be set to zero. When they are relatively prime, there is a choice, it is possible to either set one or both to zero. This in turn causes one or both of the center two W terms in (15) to become unity.

An example of the Cooley-Tukey radix-4 FFT for a length-16 DFT uses the type-two map with $K_1 = 4$, $K_2 = 1$, $K_3 = 1$, $K_4 = 4$ giving

$$n = 4n_1 + n_2 \tag{17}$$

$$k = k_1 + 4k_2 \tag{18}$$

The residue reduction in (6) is not needed here since n does not exceed N as n_1 and n_2 take on their values. Since, in this example, the factors of N have a common factor, only one of the conditions in (16) can hold and, therefore, (15) becomes

$$\hat{C}\left(k_1, k_2\right) = C\left(k\right) = \sum_{n_2=0}^{3} \sum_{n_1=0}^{3} x\left(n\right) \; W_4^{n_1 k_1} \; W_{16}^{n_2 k_1} \; W_4^{n_2 k_2} \tag{19}$$

Note the definition of W_N in (3) allows the simple form of $W_{16}^{K_1 K_3} = W_4$

This has the form of a two-dimensional DFT with an extra term W_{16}, called a "twiddle factor". The inner sum over n_1 represents four length-4 DFTs, the W_{16} term represents 16 complex multiplications, and the outer sum over n_2 represents another four length-4 DFTs. This choice of the K_i "uncouples" the calculations since the first sum over n_1 for $n_2 = 0$ calculates the DFT of the first row of the data array $\hat{x}(n_1, n_2)$, and those data values are never needed in the succeeding row calculations. The row calculations are independent, and examination of the outer sum shows that the column calculations are likewise independent. This is illustrated in Figure 1.

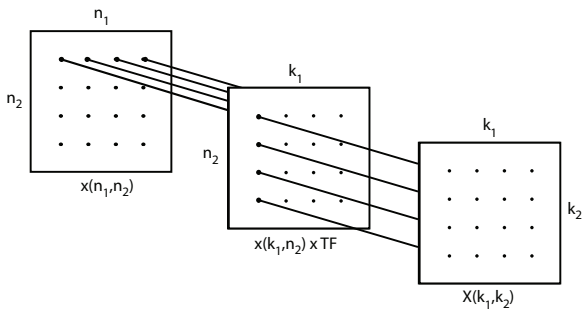

Figure 1: Uncoupling of the Row and Column Calculations (Rectangles are Data Arrays)

The left 4-by-4 array is the mapped input data, the center array has the rows transformed, and the right array is the DFT array. The row DFTs and the column DFTs are independent of each other. The twiddle factors (TF) which are the center W in (19), are the multiplications which take place on the center array of Figure 1.

This uncoupling feature reduces the amount of arithmetic required and allows the results of each row DFT to be written back over the input data locations, since that input row will not be needed again. This is called "in-place" calculation and it results in a large memory requirement savings.

An example of the type-two map used when the factors of N are relatively prime is given for $N = 15$ as

$$n = 5n_1 + n_2 \tag{20}$$

$$k = k_1 + 3k_2 \tag{21}$$

The residue reduction is again not explicitly needed. Although the factors 3 and 5 are relatively prime, use of the type-two map sets only one of the terms in (16) to zero. The DFT in (15) becomes

$$X = \sum_{n_2=0}^{4} \sum_{n_1=0}^{2} x \; W_3^{n_1 k_1} \; W_{15}^{n_2 k_1} \; W_5^{n_2 k_2} \tag{22}$$

which has the same form as (19), including the existence of the twiddle factors (TF). Here the inner sum is five length-3 DFTs, one for each value of k_1. This is illustrated in (2) where the rectangles are the 5 by 3 data arrays and the system is called a "mixed radix" FFT.

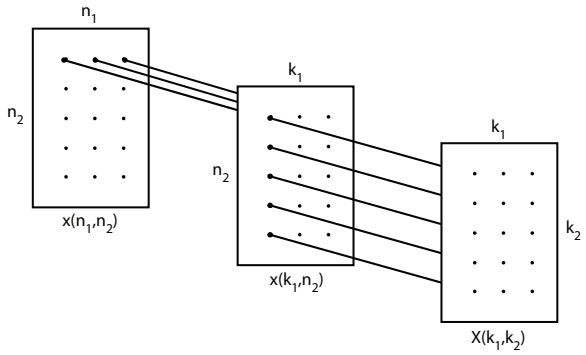

Figure 2: Uncoupling of the Row and Column Calculations (Rectangles are Data Arrays)

An alternate illustration is shown in Figure 3 where the rectangles are the short length 3 and 5 DFTs.

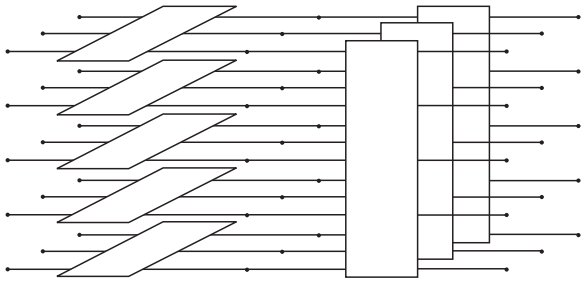

Figure 3: Uncoupling of the Row and Column Calculations (Rectangles are Short DFTs)

The type-one map is illustrated next on the same length-15 example. This time the situation of (7) with the "and" condition is used in (10) using an index map of

$$n = 5n_1 + 3n_2 \tag{23}$$

and

$$k = 10k_1 + 6k_2 \tag{24}$$

The residue reduction is now necessary. Since the factors of N are relatively prime and the type-one map is being used, both terms in (16) are zero, and (15) becomes

$$\hat{X} = \sum_{n_2=0}^{4} \sum_{n_1=0}^{2} \hat{x} \; W_3^{n_1 k_1} W_5^{n_2 k_2} \tag{25}$$

which is similar to (22), except that now the type-one map gives a pure two-dimensional DFT calculation with no TFs, and the sums can be done in either order. Figures Figure 2 and Figure 3 also describe this case but now there are no Twiddle Factor multiplications in the center and the resulting system is called a "prime factor algorithm" (PFA).

The purpose of index mapping is to improve the arithmetic efficiency. For example a direct calculation of a length-16 DFT requires 16^2 or 256 real multiplications (recall, one complex multiplication requires 4 real multiplications and 2 real additions) and an uncoupled version requires 144. A direct calculation of a length-15 DFT requires 225 multiplications but with a type-two map only 135 and with a type-one map, 120. Recall one complex multiplication requires four real multiplications and two real additions.

Algorithms of practical interest use short DFT's that require fewer than N^2 multiplications. For example, length-4 DFTs require no multiplications and, therefore, for the length-16 DFT, only the TFs must be calculated. That calculation uses 16 multiplications, many fewer than the 256 or 144 required for the direct or uncoupled calculation.

The concept of using an index map can also be applied to convolution to convert a length $N = N_1 N_2$ one-dimensional cyclic convolution into a N_1 by N_2 two-dimensional cyclic convolution [3], [1]. There is no savings of arithmetic from the mapping alone as there is with the DFT, but savings can be obtained by using special short algorithms along each dimension. This is discussed in Algorithms for Data with Restrictions[1] .

2 In-Place Calculation of the DFT and Scrambling

Because use of both the type-one and two index maps uncouples the calculations of the rows and columns of the data array, the results of each short length N_i DFT can be written back over the data as it will not be needed again after that particular row or column is transformed. This is easily seen from Figures Figure 1, Figure 2, and Figure 3 where the DFT of the first row of $x(n_1, n_2)$ can be put back over the data rather written into a new array. After all the calculations are finished, the total DFT is in the array of the original data. This gives a significant memory savings over using a separate array for the output.

Unfortunately, the use of in-place calculations results in the order of the DFT values being permuted or scrambled. This is because the data is indexed according to the input map (6) and the results are put into the same locations rather than the locations dictated by the output map (12). For example with a length-8 radix-2 FFT, the input index map is

$$n = 4n_1 + 2n_2 + n_3 \tag{26}$$

which to satisfy (16) requires an output map of

$$k = k_1 + 2k_2 + 4k_3 \tag{27}$$

The in-place calculations will place the DFT results in the locations of the input map and these should be reordered or unscrambled into the locations given by the output map. Examination of these two maps shows the scrambled output to be in a "bit reversed" order.

For certain applications, this scrambled output order is not important, but for many applications, the order must be unscrambled before the DFT can be considered complete. Because the radix of the radix-2 FFT is the same as the base of the binary number representation, the correct address for any term is found by reversing the binary bits of the address. The part of most FFT programs that does this reordering is called a bit-reversed counter. Examples of various unscramblers are found in [8], [5] and in the appendices.

The development here uses the input map and the resulting algorithm is called "decimation-in-frequency". If the output rather than the input map is used to derive the FFT algorithm so the correct output order is obtained, the input order must be scrambled so that its values are in locations specified by the output map rather than the input map. This algorithm is called "decimation-in-time". The scrambling is the same bit-reverse counting as before, but it precedes the FFT algorithm in this case. The same process of a post-unscrambler or pre-scrambler occurs for the in-place calculations with the type-one maps. Details can be

[1] "Algorithms for Data with Restrictions" <http://cnx.org/content/m16338/latest/>

found in [5], [4]. It is possible to do the unscrambling while calculating the FFT and to avoid a separate unscrambler. This is done for the Cooley-Tukey FFT in [11] and for the PFA in [5], [4], [17].

If a radix-2 FFT is used, the unscrambler is a bit-reversed counter. If a radix-4 FFT is used, the unscrambler is a base-4 reversed counter, and similarly for radix-8 and others. However, if for the radix-4 FFT, the short length-4 DFTs (butterflies) have their outputs in bit-revered order, the output of the total radix-4 FFT will be in bit-reversed order, not base-4 reversed order. This means any radix-2^n FFT can use the same radix-2 bit-reversed counter as an unscrambler if the proper butterflies are used.

3 Efficiencies Resulting from Index Mapping with the DFT

In this section the reductions in arithmetic in the DFT that result from the index mapping alone will be examined. In practical algorithms several methods are always combined, but it is helpful in understanding the effects of a particular method to study it alone.

The most general form of an uncoupled two-dimensional DFT is given by

$$X\left(k1,k2\right)=\sum_{n_2=0}^{N_2-1}\left\{\sum_{n_1=0}^{N_1-1}x\left(n_1,n_2\right)\ \ f_1\left(n_1,n_2,k_1\right)\right\}\ f_2\left(n_2,k_1,k_2\right) \tag{28}$$

where the inner sum calculates N_2 length-N_1 DFT's and, if for a type-two map, the effects of the TFs. If the number of arithmetic operations for a length-N DFT is denoted by $F\left(N\right)$, the number of operations for this inner sum is $F=N_2F\left(N_1\right)$. The outer sum which gives N_1 length-N_2 DFT's requires $N_1F\left(N_2\right)$ operations. The total number of arithmetic operations is then

$$F=N_2F\left(N_1\right)+N_1F\left(N_2\right) \tag{29}$$

The first question to be considered is for a fixed length N, what is the optimal relation of N_1 and N_2 in the sense of minimizing the required amount of arithmetic. To answer this question, $N1$ and N_2 are temporarily assumed to be real variables rather than integers. If the short length-N_i DFT's in (28) and any TF multiplications are assumed to require N_i^2 operations, i.e. $F\left(N_i\right)=N_i^2$, "Efficiencies Resulting from Index Mapping with the DFT" (Section 3: Efficiencies Resulting from Index Mapping with the DFT) becomes

$$F=N_2N_1^2+N_1N_2^2=N\left(N_1+N_2\right)=N\left(N_1+NN_1^{-1}\right) \tag{30}$$

To find the minimum of F over N_1, the derivative of F with respect to N_1 is set to zero (temporarily assuming the variables to be continuous) and the result requires $N_1=N_2$.

$$dF/dN_1=0\quad =>\quad N_1=N_2 \tag{31}$$

This result is also easily seen from the symmetry of N_1 and N_2 in $N=N_1N_2$. If a more general model of the arithmetic complexity of the short DFT's is used, the same result is obtained, but a closer examination must be made to assure that $N_1=N_2$ is a global minimum.

If only the effects of the index mapping are to be considered, then the $F\left(N\right)=N^2$ model is used and (31) states that the two factors should be equal. If there are M factors, a similar reasoning shows that all M factors should be equal. For the sequence of length

$$N=R^M \tag{32}$$

there are now M length-R DFT's and, since the factors are all equal, the index map must be type two. This means there must be twiddle factors.

In order to simplify the analysis, only the number of multiplications will be considered. If the number of multiplications for a length-R DFT is $F(R)$, then the formula for operation counts in (30) generalizes to

$$F = N\sum_{i=1}^{M} F(N_i)/N_i = NMF(R)/R \tag{33}$$

for $N_i = R$

$$F = NlnR(N)F(R)/R = (NlnN)(F(R)/(RlnR)) \tag{34}$$

This is a very important formula which was derived by Cooley and Tukey in their famous paper [6] on the FFT. It states that for a given R which is called the radix, the number of multiplications (and additions) is proportional to $NlnN$. It also shows the relation to the value of the radix, R.

In order to get some idea of the "best" radix, the number of multiplications to compute a length-R DFT is assumed to be $F(R) = R^x$. If this is used with (34), the optimal R can be found.

$$dF/dR = 0 \quad => \quad R = e^{1/(x-1)} \tag{35}$$

For $x = 2$ this gives $R = e$, with the closest integer being three.

The result of this analysis states that if no other arithmetic saving methods other than index mapping are used, and if the length-R DFT's plus TFs require $F = R^2$ multiplications, the optimal algorithm requires

$$F = 3Nlog_3N \tag{36}$$

multiplications for a length $N = 3^M$ DFT. Compare this with N^2 for a direct calculation and the improvement is obvious.

While this is an interesting result from the analysis of the effects of index mapping alone, in practice, index mapping is almost always used in conjunction with special algorithms for the short length-N_i DFT's in (15). For example, if $R = 2$ or 4, there are no multiplications required for the short DFT's. Only the TFs require multiplications. Winograd (see Winorad's Short DFT Algorithms[2]) has derived some algorithms for short DFT's that require $O(N)$ multiplications. This means that $F(N_i) = KN_i$ and the operation count F in "Efficiencies Resulting from Index Mapping with the DFT" (Section 3: Efficiencies Resulting from Index Mapping with the DFT) is independent of N_i. Therefore, the derivative of F is zero for all N_i. Obviously, these particular cases must be examined.

4 The FFT as a Recursive Evaluation of the DFT

It is possible to formulate the DFT so a length-N DFT can be calculated in terms of two length-(N/2) DFTs. And, if $N = 2^M$, each of those length-(N/2) DFTs can be found in terms of length-(N/4) DFTs. This allows the DFT to be calculated by a recursive algorithm with M recursions, giving the familiar order $Nlog(N)$ arithmetic complexity.

Calculate the even indexed DFT values from (1) by:

$$C(2k) = \sum_{n=0}^{N-1} x(n)\ W_N^{2nk} = \sum_{n=0}^{N-1} x(n)\ W_{N/2}^{nk} \tag{37}$$

$$C(2k) = \sum_{n=0}^{N/2-1} x(n)\ W_N^{2nk} + \sum_{n=N/2}^{N-1} x(n)\ W_{N/2}^{nk} \tag{38}$$

$$C(2k) = \sum_{n=0}^{N/2-1} \{x(n) + x(n+N/2)\}\ W_{N/2}^{nk} \tag{39}$$

[2] "Winograd's Short DFT Algorithms" <http://cnx.org/content/m16333/latest/>

and a similar argument gives the odd indexed values as:

$$C(2k+1) = \sum_{n=0}^{N/2-1} \{x(n) \ - \ x(n+N/2)\} \ W_N^n \ W_{N/2}^{nk} \tag{40}$$

Together, these are recursive DFT formulas expressing the length-N DFT of $x(n)$ in terms of length-N/2 DFTs:

$$C(2k) = \mathrm{DFT}_{N/2}\{x(n) \ + \ x(n+N/2)\} \tag{41}$$

$$C(2k+1) = \mathrm{DFT}_{N/2}\{[x(n) \ - \ x(n+N/2)] \, W_N^n\} \tag{42}$$

This is a "decimation-in-frequency" (DIF) version since it gives samples of the frequency domain representation in terms of blocks of the time domain signal.

A recursive Matlab program which implements this is given by:

```
        function c = dftr2(x)
% Recursive Decimation-in-Frequency FFT algorithm, csb 8/21/07
L = length(x);
if L > 1
     L2 = L/2;
     TF = exp(-j*2*pi/L).^[0:L2-1];
     c1 = dftr2( x(1:L2) + x(L2+1:L));
     c2 = dftr2((x(1:L2) - x(L2+1:L)).*TF);
     cc = [c1';c2'];
     c = cc(:);
else
     c  = x;
end
```

Listing 1: DIF Recursive FFT for $N = 2^M$

A DIT version can be derived in the form:

$$C(k) = \mathrm{DFT}_{N/2}\{x(2n)\} \ + \ W_N^k \mathrm{DFT}_{N/2}\{x(2n+1)\} \tag{43}$$

$$C(k+N/2) = \mathrm{DFT}_{N/2}\{x(2n)\} \ - \ W_N^k \mathrm{DFT}_{N/2}\{x(2n+1)\} \tag{44}$$

which gives blocks of the frequency domain from samples of the signal.

A recursive Matlab program which implements this is given by:

```
        function c = dftr(x)
% Recursive Decimation-in-Time FFT algorithm, csb
L = length(x);
if L > 1
     L2 = L/2;
     ce = dftr(x(1:2:L-1));
     co = dftr(x(2:2:L));
     TF = exp(-j*2*pi/L).^[0:L2-1];
     c1 = TF.*co;
     c  = [(ce+c1), (ce-c1)];
else
     c  = x;
end
```

Listing 2: DIT Recursive FFT for $N = 2^M$

Similar recursive expressions can be developed for other radices and and algorithms. Most recursive programs do not execute as efficiently as looped or straight code, but some can be very efficient, e.g. parts of the FFTW.

Note a length-2^M sequence will require M recursions, each of which will require $N/2$ multiplications. This give the $Nlog\left(N\right)$ formula that the other approaches also derive.

References

[1] R. C. Agarwal and J. W. Cooley. New algorithms for digital convolution. *IEEE Trans. on ASSP*, 25(2):3928211;410, October 1977.

[2] Richard E. Blahut. *Fast Algorithms for Digital Signal Processing.* Addison-Wesley, Reading, Mass., 1985.

[3] C. S. Burrus. Index mapping for multidimensional formulation of the dft and convolution. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-25(3):2398211;242, June 1977.

[4] C. S. Burrus and P. W. Eschenbacher. An in8211;place, in8211;order prime factor fft algorithm. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 29(4):8068211;817, August 1981. Reprinted in it DSP Software, by L.R. Morris, 1983.

[5] C. S. Burrus and T. W. Parks. *DFT/FFT and Convolution Algorithms.* John Wiley & Sons, New York, 1985.

[6] J. W. Cooley and J. W. Tukey. An algorithm for the machine calculation of complex fourier series. *Math. Computat.*, 19:2978211;301, 1965.

[7] Douglas F. Elliott, editor. *Handbook of Digital Signal Processing.* Academic Press, San Diego, CA, 1987. Chapter 7 on FFT by Elliott.

[8] B. Gold and C. M. Rader. *Digital Processing of Signals.* McGraw-Hill, New York, 1969.

[9] I. J. Good. Interaction algorithm and practical fourier analysis. *J. Royal Statist. Soc., B*, 20:3618211;372, 1958. Addendum: vol. 22, 1960, pp 3728211;375.

[10] Markus Hegland and W. W. Wheeler. Linear bijections and the fast fourier transform. *Applicable Algebra in Engineering, Communication and Computing*, 8(2):1438211;163, 1997.

[11] H. W. Johnson and C. S. Burrus. An in-order, in-place radix-2 fft. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, page 28A.2.18211;4, San Diego, CA, March 1984.

[12] D. P-K. Lun and W-C. Siu. An analysis for the realization of an in-place and in-order prime factor algorithm. *IEEE Transactions on Signal Processing*, 41(7):23628211;2370, July 1993.

[13] J. H. McClellan and C. M. Rader. *Number Theory in Digital Signal Processing*. Prentice-Hall, Englewood Cliffs, NJ, 1979.

[14] A. V. Oppenheim and R. W. Schafer. *Discrete-Time Signal Processing*. Prentice-Hall, Englewood Cliffs, NJ, second edition, 1999. Earlier editions in 1975 and 1989.

[15] T. W. Parks and C. S. Burrus. *Digital Filter Design*. John Wiley & Sons, New York, 1987.

[16] L. R. Rabiner and C. M. Rader, editors. *Digital Signal Processing, selected reprints*. IEEE Press, New York, 1972.

[17] J.H. Rothweiler. Implementation of the in-order prime factor fft algorithm. *IEEE TRANS. ON ASSP*, 30:105–107, February 1982.

[18] James C. Schatzman. Index mapping for the fast fourier transform. *IEEE Transactions on Signal Processing*, 44(3):7178211;719, March 1996.

[19] S. Winograd. On computing the discrete fourier transform. *Proc. National Academy of Sciences, USA*, 73:10068211;1006, April 1976.