Efficient Computation of the DFT with Only a Subset of Input or Output Points

Henrik V. Sorensen, *Member, IEEE,* and C. Sidney Burrus, *Fellow, IEEE*

Abstract—**The standard FFT algorithms inherently assume that the length of the input and output sequences are equal. In practice this is not always an accurate assumption, and this paper discusses ways of efficiently computing the DFT, when the number of input and output data points differ.**

The two problems, whether the length of the input or output sequence is reduced, can be found to be "duals" of each other, and the same methods can, to a large extent, be used to solve both. The algorithms utilize the redundancy in the input or output to reduce the number of operations below those of the FFT algorithms. This paper briefly discusses the well-known pruning method and introduces a new efficient algorithm called transform decomposition. This new algorithm is based on a mixture of a standard FFT algorithm and Horner's polynomial evaluation scheme equivalent to the one in Goertzel's algorithm. It is shown to require fewer operations than pruning and also to be more flexible than pruning. The algorithm works for both power of two and prime-factor algorithms, as well as for real input data.

I. Introduction

THE discrete Fourier transform (DFT) is probably the single most important tool in digital signal processing (DSP), a fact that can be accredited to the discovery of the radix-2 fast Fourier transform (FFT) by Cooley and Tukey at IBM in 1965 [4]. The radix-2 algorithm lead to an intense research effort by several people to develop higher radix algorithms [1], mixed-radix [13], prime-factor [8] (PFA), Winograd (WFTA) [20], and most recently the split-radix [16] Fourier transform algorithms. These algorithms make different assumptions about the transform length, N, but they all assume that the input and the output sequence lengths are equal. For most applications this is a reasonable assumption, but there remain some applications where the number of input and output points differ significantly, such as computing a high-resolution spectrum of eigenvalues in array processing or computing the transition-region (between the passband and stopband) response in a filter design.

This paper presents ways of utilizing the redundancies, due to a reduction in either the number of input or output points, of the standard FFT algorithms mentioned above. It briefly discusses well-known pruning algorithms [9] to develop a foundation for comparison and to present a comparative study of these algorithms. The two problems, whether dealing with only a subset of input or a subset of output points, are "duals" of each other in much the same way as the decimation-in-time and decimation-in-frequency FFT algorithms are "duals" of each other. The paper will hence focus on the case of a subset of output points and draw parallels to the case of a subset of inputs. Let $x(n)$ be a length N sequence. Its DFT, $X(k)$, is also a sequence of length N defined as

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

where $W_N = e^{-j(2\pi/N)}$.

Manuscript received March 16, 1989; revised February 8, 1991. This work was supported in part by Texas Instruments and the University of Pennsylvania, and by NSF NCR90-16165.

H. V. Sorensen is with the Department of Electrical Engineering, University of Pennsylvania, Philadelphia, PA 19104.

C. S. Burrus is with the Department of Electrical and Computer Engineering, Rice University, Houston, TX 77251.

IEEE Log Number 9206041.

II. Computation of a Subset of Output Points

Sometimes only a narrow spectrum is of interest, but the resolution within that band has to be very high. Hence the DFT has N input values, but fewer than N outputs are needed. The first algorithm useful for computing only a subset of the output points was Goertzel's algorithm [7]. Given N input points, Goertzel's algorithm requires $O(8N)$ real operations per output point, but since there exist FFT algorithms that require $O(4N \log N)$ real operations to compute all the N outputs [16], Goertzel's algorithm only saves operations over the FFT up to $1/2 \log N$ output points. Another algorithm, which gives a slight improvement over Goertzel's algorithm, is described by Boncelet [2]. His method requires on the average $O(8N/3)$ real operations per output point, which is still much more than the following methods.

A. Pruning

The pruning method was first devised by Markel [9] and later improved by Skinner [14], Sreenivas and Rao [18], and Nagai [10], and is also sometimes called the zoom-FFT [6]. Pruning is a modification of the standard one-butterfly radix-2 FFT described in [3]. Fig. 1 shows how this pruning scheme works. Assuming that only $X(0)$ and $X(1)$ are of interest, only the solid edges in the flow graph need to be computed, while the grey edges can be "pruned" away. Given that only L output points are needed (in Fig. 1 L is two), only the first L values in each group of butterflies are required. This can easily be im-

1053-587X/93$03.00 © 1993 IEEE

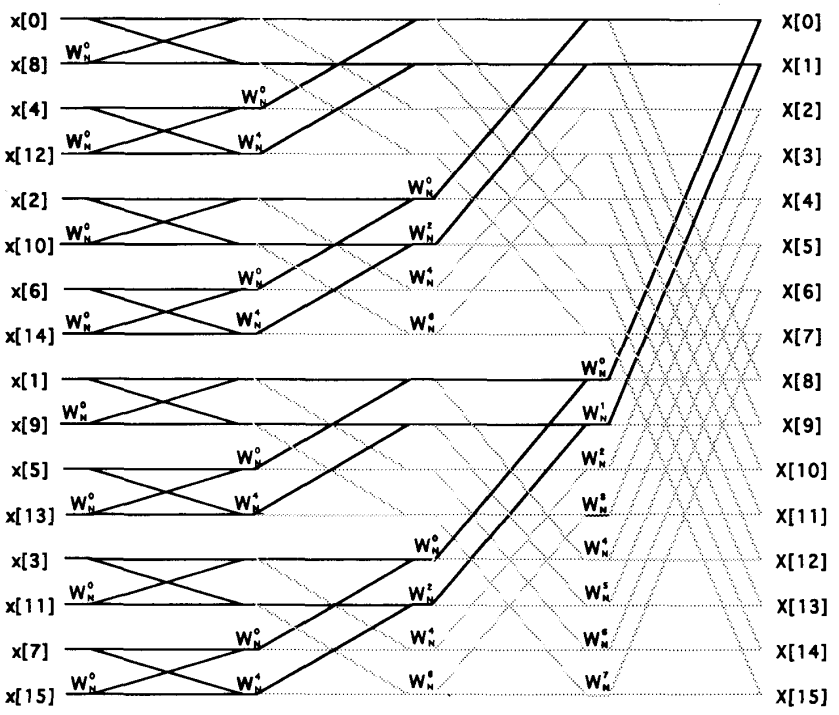


Fig. 1. Length 16 pruned FFT for a subset of output points.

plemented by modifying one of the three "loop limits" in a regular radix-2 FFT program. In the first stages L might be greater than the actual number of butterflies in each group, and hence the new limit should be the minimum of the regular limit from the radix-2 program and L. As can be seen from Fig. 1, the butterflies in the last stages are only half butterflies, while they are the "normal" ones in the first stage. Often pruning programs ignore this fact and just compute regular butterflies instead, which gives a more compact program, but increases the number of operations. However, to take advantage of these half butterflies a conditional statement (an *if* statement) often has to be introduced into the program, and the time to execute this often exceeds the savings obtained by the fewer operations.

By also shifting the twiddle factor in the program [10] it is possible to get a band that does not start at $X(0)$, but can start anywhere. Multiplying all the twiddle factors by W_N^J the L output values will be $X(J)$, $X(J+1)$, $\cdots$, $X(J+L-1)$ instead of $X(0)$, $X(1)$, $\cdots$, $X(L-1)$.

To compute L out of N DFT points, the regular pruning program requires

$$\#\text{MUL}_{\text{PRUNE}} = 2N \lfloor \log_2 L \rfloor + 2N - 4L + \frac{2NL}{2^{\lfloor \log_2 L \rfloor}} \tag{2}$$

multiplications and

$$\#\text{ADD}_{\text{PRUNE}} = 3N \lfloor \log_2 L \rfloor + 3N - 6L + \frac{3NL}{2^{\lfloor \log_2 L \rfloor}} \tag{3}$$

additions, where the function $\lfloor\ \rfloor$ is the integer part of its argument. They reduce to

$$\#\text{MUL}_{\text{PRUNE}} = 2N \log_2 L - 4L + 4N \tag{4}$$

multiplications and

$$\#\text{ADD}_{\text{PRUNE}} = 3N \log_2 L - 6L + 6N \tag{5}$$

addition when L is a power of two.

If no offset is needed (the outputs needed are $X(0)$, $X(1)$, $\cdots$, $X(L-1)$), a two butterfly pruned FFT can be obtained by separately computing the $X(0)$ term, which has a trivial twiddle factor. The operation counts for the two butterfly pruned FFT are

$$\#\text{MUL}_{\text{2BF-NO-OFFSET}} = 2N \lfloor \log_2 L \rfloor - 4L - 2N + 4 + \frac{2NL}{2^{\lfloor \log_2 L \rfloor}} \tag{6}$$

multiplications and

$$\#\text{ADD}_{\text{2BF-NO-OFFSET}} = 3N \lfloor \log_2 L \rfloor - 6L + N + 2 + \frac{3NL}{2^{\lfloor \log_2 L \rfloor}} \tag{7}$$

additions. Without making assumptions on the value of L, the optimal pruned radix-2 FFT is a 5 butterfly version, which takes the same number of multiplications as the 2 butterfly version and

$$\#\text{ADD}_{\text{5BF-NO-OFFSET}} = 3N \lfloor \log_2 L \rfloor - 4L - N + 2 + \frac{3NL}{2^{\lfloor \log_2 L \rfloor}} \tag{8}$$

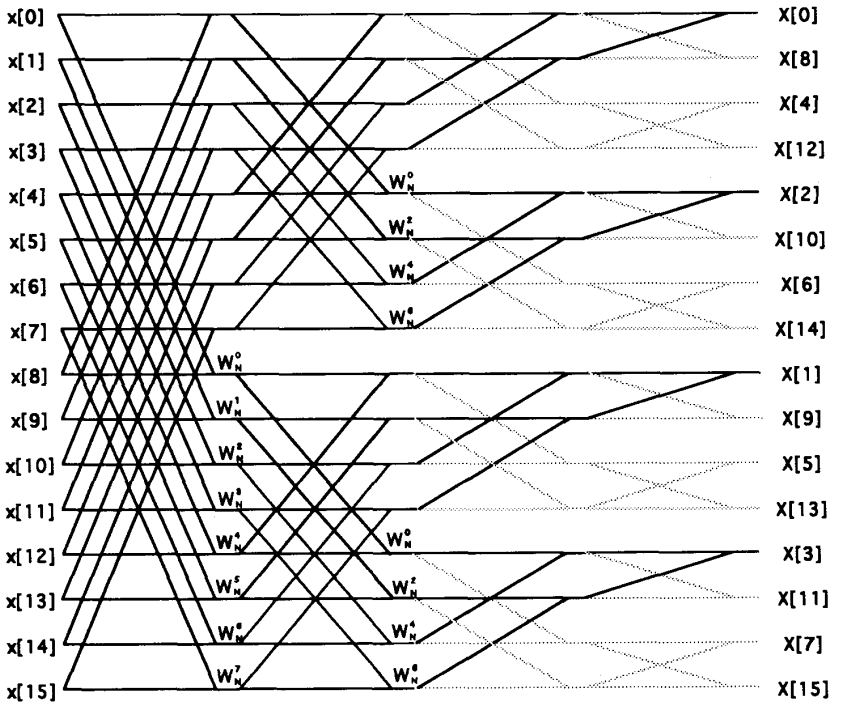


Fig. 2. Skinner's pruned FFT algorithm modified for a subset of output points.

additions. It uses both full and half butterflies and hence requires an if statement inside the outside (first) loop index and is thus not a very practical choice.

If L is restricted to be a power of two, Skinner developed an algorithm, which is shown in Fig. 2, that is slightly more efficient than the 5 butterfly algorithm above. It is achieved by pruning a decimation-in-frequency algorithm instead of the decimation-in-time that Markel's method is based on. In Skinner's algorithm, the last $\log_2 L$ stages does not contain any twiddle factors, but only additions. The algorithm can be found to require

$$\#\mathrm{MUL}_{\mathrm{SKINNER}} = 2N \log_2 L \tag{9}$$

multiplications and

$$\#\mathrm{ADD}_{\mathrm{SKINNER}} = 3N \log_2 L + 2N - 2L \tag{10}$$

addition. Fig. 3 shows the total number of operations for computing a subset of output points of a length 512 DFT for the pruned FFT algorithms discussed in this section. Skinner's algorithm and the 5 butterfly algorithm are indistinguishable on the chosen scale, but it is apparent that they do not save that many operations over the 2 butterfly version.

It is possible to prune higher radix and split-radix algorithms, but in practice they become very inefficient. For example, a radix-4 algorithm would need four different radix-4 butterflies (producing 1, 2, 3, or 4 outputs) to efficiently compute the FFT and selecting between these four butterflies consumes the savings obtained by the higher radix algorithm. If the transform size N and the number of input (or output) elements are fixed, it is possible to generate efficient code for high- or split-radix prunned algorithms, but if the output (or input) points are not known *a priori*, the selection of butterflies have to be done at run time and will require conditional (*if* statements) in the code.

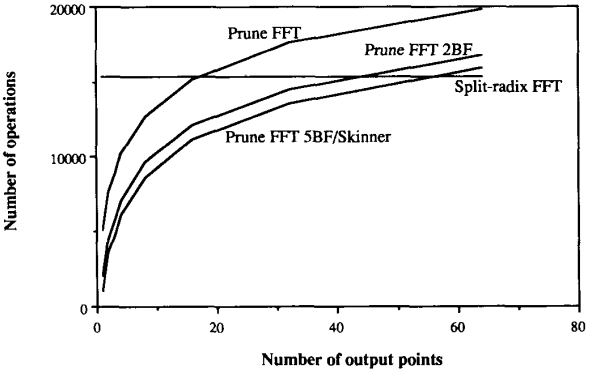


Fig. 3. Comparison of pruned FFT's for length 512 transforms.

B. Transform Decomposition

A new method, transform decomposition, for computing only a subset of output points will now be introduced [15]. It uses a mixture of a Cooley–Tukey FFT and a computational structure similar to Goertzel's algorithm.

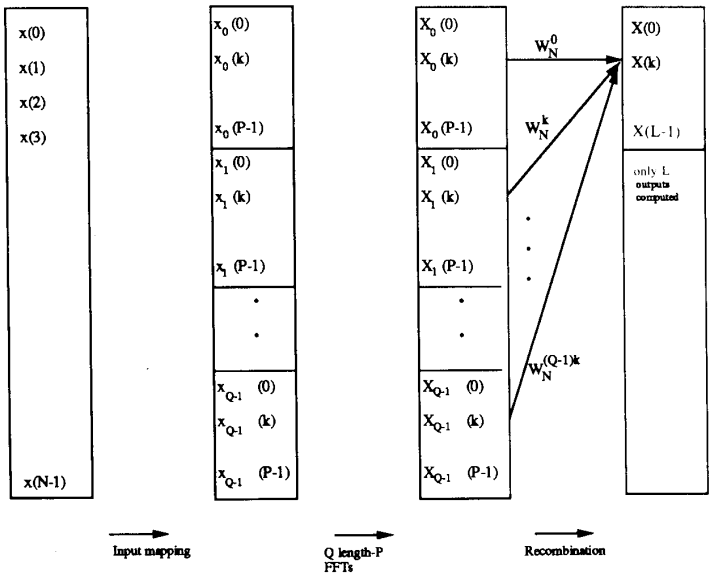


Fig. 4. Block diagram over transform decomposition for a subset of outputs.

It is shown to be both more efficient and more flexible than pruning.

The definition of the DFT in (1) was

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk} \qquad k = 0, 1, \cdots, N-1. \quad (11)$$

Assume again that only L output points are needed and that there exists a P such that P divides N and define $Q = N/P$. Using the variable substitution

$$n = Qn_1 + n_2 \qquad \begin{matrix} n_1 = 0, 1, \cdots, P-1 \\ n_2 = 0, 1, \cdots, Q-1 \end{matrix} \quad (12)$$

we can rewrite the DFT as follows:

$$X(k) = \sum_{n_2=0}^{Q-1} \sum_{n_1=0}^{P-1} x(n_1 Q + n_2) W_N^{(n_1 Q + n_2)k} \quad (13)$$

$$= \sum_{n_2=0}^{Q-1} \left[\sum_{n_1=0}^{P-1} x(n_1 Q + n_2) W^{n_1 \langle k \rangle_P} \right] W_N^{n_2 k} \quad (14)$$

where $\langle\ \rangle_P$ denotes reduction modulo P, and k takes on any L consecutive values between 0 and $N-1$. Breaking this up into two equations

$$X(k) = \sum_{n_2=0}^{Q-1} X_{n_2}(\langle k \rangle_P) W_N^{n_2 k} \quad (15)$$

where

$$X_{n_2}(j) = \sum_{n_1=0}^{P-1} x(n_1 Q + n_2) W_P^{n_1 j} \quad (16)$$

$$= \sum_{n_1=0}^{P-1} x_{n_2}(n_1) W_P^{n_1 j} \qquad j = 0, 1, \cdots, P-1 \quad (17)$$

and

$$x_{n_2}(n_1) = x(n_1 Q + n_2). \quad (18)$$

The sum in (17) can be recognized as a length P DFT, and it can be computed efficiently using any FFT algorithm. This is a great advantage of the transform decomposition method over the pruning method, which in practice, due to the structure of the pruning scheme, must use a simple one butterfly radix-2 FFT, which requires far more operations than the split-radix or prime-factor algorithms.

Inspecting (17), it can be seen that the sequence over which the DFT has to be computed is two dimensional and hence depends on n_2. Thus a DFT has to be computed for each different value of n_2, and hence there are Q such length P DFT's. The output of the DFT's are recombined using (15) which can be computed directly using Q multiplications and $Q-1$ additions per output point or a total of QL complex multiplications, each requiring 4 real additions and 2 additions and $L(Q-1)$ complex additions each requiring 2 real additions. Fig. 4 shows how this method works to compute the first L out of N DFT points. Rewriting (15) as

$$X(k) = X_0(\langle k \rangle_P) + \sum_{n_2-1}^{Q-1} X_{n_2}(\langle k \rangle_P) W_N^{n_2 k} \quad (19)$$

the number of complex multiplications is reduced by L to $L(Q-1)$.

As mentioned earlier, the algorithm works both for N being a power of two and for N being a product of primes, but since there does not exist a formula for the prime-factor algorithms, the following discussion is restricted to the power of two case. If N is a power of two (and hence P is a power of two) the FFT's can be computed using a

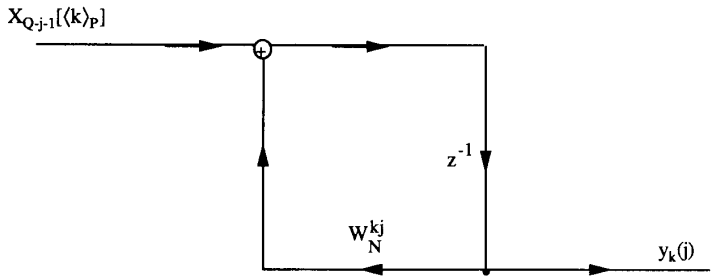


Fig. 5. Flow graph of first-order network to compute (26).

three butterfly split-radix FFT [16], which requires the lowest number of operations of any known power of two FFT:

$$\#\mathrm{MUL}_{\mathrm{FFT}}(N) = N \log_2 N - 3N + 4 \tag{20}$$

multiplications and

$$\#\mathrm{ADD}_{\mathrm{FFT}}(N) = 3N \log_2 N - 3N + 4 \tag{21}$$

additions to compute a length N DFT. The total number of operations for computing the transform decomposition algorithm, given that N is a power of two, is hence

$$\#\mathrm{MUL}_{\mathrm{TD}} = N \log_2 P - 3N + 4(L+1)\frac{N}{P} - 4L \tag{22}$$

multiplications and

$$\#\mathrm{ADD}_{\mathrm{TD}} = 3N \log_2 P - 3N + 4(L+1)\frac{N}{P} - 4L \tag{23}$$

additions.

It is possible to lower the number of operations required to compute (15) even further using a technique similar to Goertzel's algorithm [5], [12], [7]. Just like Goertzel's algorithm, this scheme is based on Horner's rule for evaluation of a polynomial at a single point [15]. To see this, rewrite (15) as follows:

$$X(k) = \sum_{n_2=0}^{Q-1} X_{n_2}(\langle k \rangle_P)(W_N^k)^{n_2} \tag{24}$$

$$= \sum_{m=0}^{Q-1} X_{Q-m-1}(\langle k \rangle_P)(W_N^k)^{Q-m-1} \tag{25}$$

with the variable substitution $m = Q - n_2 - 1$. Now define

$$y_k(j) = \sum_{m=0}^{j-1} X_{Q-m-1}(\langle k \rangle_P)(W_N^k)^{j-m-1} \tag{26}$$

from which we can find $X(k)$ as

$$X(k) = y_k(j)\big|_{j=Q}. \tag{27}$$

Equation (26) can be recognized as a shifted cyclic convolution between the sequence $X_{Q-j-1}(\langle k \rangle_P)$ and $(W_N^k)^{j-1}$ in the variable j (similar to the derivation of the Goertzel algorithm in [11]) and hence $y_k(j)$ can be viewed as the output of a system with impulse response $(W_N^k)^{j-1}$ driven by the input $X_{Q-j-1}(\langle k \rangle_P)$.

Fig. 5 shows a flow graph that implements (26), but a quick analysis will show that this implementation requires 4 multiplications per iteration assuming the input is complex, and hence requires the same amount of operations as a direct implementation of (15).

The transfer function of the system in Fig. 5 can be found as

$$H_k(z) = \frac{z^{-1}}{1 - z^{-1} W_N^k} \tag{28}$$

which can be rewritten as

$$H_k(z) = \frac{z^{-1}(1 - z^{-1} W_N^{-k})}{(1 - z^{-1} W_N^k)(1 - z^{-1} W_N^{-k})} \tag{29}$$

$$= \frac{z^{-1}(1 - z^{-1} W_N^{-k})}{(1 - 2\cos\left(\frac{2\pi k}{N}\right) z^{-1} + z^{-2}} \tag{30}$$

This last equation can be implemented using the flow graph in Fig. 6. Assuming the input is complex, each iteration only takes two multiplications since the multiplication by -1 need not be counted. This is half of what was needed in the first-order case. Because we are only interested in $y_k(Q)$, but not the intermediate values, it can be seen that the zero of the system is only needed once, just as for the Goertzel algorithm. From these results it can easily be shown that this method requires about the same number of additions and about half the multiplications of a direct implementation of (25).

L output values are needed, and each value requires $Q - 1$ evaluations of the poles of Fig. 6 and one evaluation of the zeros. Since each evaluation of the poles requires two real multiplications and four real additions each evaluation of the zeros of the system requires four real multiplications and four real additions, we get a total of $2L(Q - 1) + 4L$ real multiplications and $4L(Q - 1) + 4L$ real additions. If considering power of two algorithms, the operations for the FFT's can be added to get

$$\#\mathrm{MUL}_{\mathrm{TD\text{-}FILT}} = N \log_2 P - 3N + 2(L+2)\frac{N}{P} + 2L \tag{31}$$

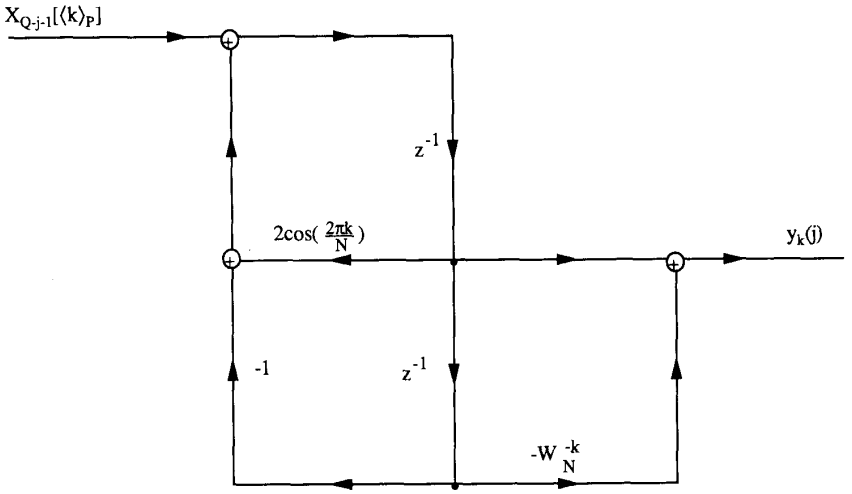


Fig. 6. Flow graph of second-order network to compute (30).

real multiplications and

$$\#\text{ADD}_{\text{TD-FILT}} = 3N \log_2 P - 3N + 4(L+1)\frac{N}{P} - 4L + 4P \quad (32)$$

real additions for this method. Fig. 7 shows a program which implements the transform decomposition method using this approach to achieve these operation counts.

Another way to view transform decomposition can be found, if it is considered in the framework of the Cooley–Tukey FFT. In the first stage of a radix-P FFT the length $N = PQ$ input sequence is converted into a two-dimensional P by Q sequence. To compute the length N DFT, length P DFT's are applied to the columns and length Q DFT's are applied to the rows. Since P and Q are not relative prime in the Cooley–Tukey FFT, a set of multiplicative factors, twiddle factors, are multiplied on the result of the column DFT's before applying the row DFT's. The Cooley–Tukey FFT applies this approach recursively to the length Q DFT's, while the transform decomposition method computes them using the flow graph in Fig. 6. This flow graph requires more operations per output point than the Cooley–Tukey method, but unlike the Cooley Tukey FFT, it is not based on all the output points being computed. Notice, if $L = P$ is a power of two, the flow-graph computation in Fig. 6 is equivalent to the computation in the last stages of a regular pruning algorithm, but requires about half the multiplications. In other words, (15) represents the computation of the half butterflies in the last three stages in Fig. 1 with $N = 8$, $L = P = 2$.

The derivation of (26) and (27) is not based on the actual values of the indices of the computed output values, i.e., does not rely on the specific values of k. Unlike the standard FFT's, efficient computation of (26) and (27) by the flow graph in Fig. 6 does not depend on combining computations for several different output points (several different k's). Hence the number of output points to be computed can be any length L subset of the N possible output points. Fig. 8 shows how Fig. 4 can be modified to compute three output points, $X(3)$, $X(4)$, and $X(7)$, out of $N = 8$ with the transform length $P = 4$. This is a very powerful result that shows that transform decomposition is not just more efficient than pruning, but also more flexible. Where pruning restricts you to L subsequent output values, transform decomposition allows any length L subset to be computed.

It still needs to be determined what values to use for the factor P. For most applications the number of output points L is given and the optimum P has to be found. The solution depends on the optimality criterion used. To minimize the total number of operations, write an expression for the total number of operations for computing L output points using Q length P transforms from (22) and (23). Then take the derivative and solving for P, it can be found that, to minimize the total number of operations, P should be chosen as

$$P_{\text{TOT-MIN-TD}} = \lfloor 2(L+1) \log_e 2 \rfloor \quad (33)$$

where $\lfloor\ \rfloor$ indicates "closest" power of two. Unfortunately, the problem is nonlinear, and hence it is not "closest" in any easily determined sense, so both the larger and the smaller possible value of P should be examined. If instead the lowest possible number of multiplications is required, P should be chosen as

$$P_{\text{MUL-MIN-TD}} = \lfloor 4(L+1) \log_e 2 \rfloor \quad (34)$$

and if the lowest possible number of additions is required

$$P_{\text{ADD-MIN-TD}} = \left\lfloor \frac{4}{3}(L+1) \log_e 2 \right\rfloor \quad (35)$$

(normally not a useful choice, but is included for completeness). Hence if the total number of operations is to be minimized, P should be chosen slightly larger (1.38

```
CC=================================================================CC
CC                                                                 CC
CC  Subroutine LOTDFT(X,Y,M,IA,L,D,E):                             CC
CC      A transform-decomposition FFT for computing a limited      CC
CC      set of output points of a length N transform.              CC
CC      The number of calculated output points is L                CC
CC      and they are indexes through the array IA.                 CC
CC      The algorith call a regular power-of-two algorithm         CC
CC      (or split-radix FFT)                                       CC
CC                                                                 CC
CC  Input/output:                                                  CC
CC      X    Array of real part of input/output (length >= N)      CC
CC      Y    Array of imaginary part of input/output (length >= N) CC
CC      M    Transform length is N=2**M                            CC
CC      IA   Index-array of desired output point                   CC
CC      L    Number of desired output points                       CC
CC      D    Work array of length >= N                             CC
CC      E    Work array of length >= N                             CC
CC                                                                 CC
CC  Calls:                                                         CC
CC      FFT - A FFT program                                        CC
CC                                                                 CC
CC  Author:                                                        CC
CC      Henrik Sorensen,  University of Pennsylvania,  Sep. 1987   CC
CC                        Arpa address: hvs@ee.upenn.edu           CC
CC  Modified:                                                      CC
CC      Henrik Sorensen,  University of Pennsylvania,  Feb. 1988   CC
CC      Henrik Sorensen,  University of Pennsylvania,  Oct. 1988   CC
CC                                                                 CC
CC      This program may be used and distributed freely as long    CC
CC      as this header is included                                 CC
CC                                                                 CC
CC=================================================================CC
      SUBROUTINE LOTDFT(X,Y,M,IA,L,D,E)
      REAL X(1),Y(1),D(1),E(1)
      REAL CC,CC2,SS,ANG,A1,A2,B1,B2,T
      INTEGER IA(1)
      INTEGER N,M,P,R,Q,J,K,L,K0
      N = 2**M
C Choose P. Can also be input to program
      R=INT(LOG(FLOAT(L))/LOG(2.0)+0.5)
      P = 2**R
      Q = N/P
      DO  30 J=0,Q-1
          DO  40 K=0,P-1
              D(J*P+K+1) = X(K*Q+J+1)
              E(J*P+K+1) = Y(K*Q+J+1)
 40       CONTINUE
          CALL FFT(D(J*P+1),E(J*P+1),P,R)
 30   CONTINUE
      DO  50 K=1,L
C The index array IA can be removed if only one output band is needed.
C The band will be: [IOFF,IOFF+1,...IOFF+L-1]
CCCCCCCCCC K0 = MOD(K+IOFF-1,P)+1
CCCCCCCCCC ANG = 6.28318530718/N*(K+IOFF-1)
          K0 = MOD(IA(K),P)+1
          ANG = 6.28318530718/N*IA(K)
          CC = COS(ANG)
          SS = SIN(ANG)
          CC2 = 2*CC
          A2 = 0
          B2 = 0
          A1 = D(K0+N-P)
          B1 = E(K0+N-P)
          DO  55 J=K0+N-2*P,K0,-P
              T  = CC2*A1 - A2 + D(J)
              A2 = A1
              A1 = T
              T  = CC2*B1 - B2 + E(J)
              B2 = B1
              B1 = T
 55       CONTINUE
          X(K) = A1 - CC*A2 + SS*B2
          Y(K) = B1 - SS*A2 - CC*B2
 50   CONTINUE
      RETURN
      END
```

Fig. 7. Program for transform decomposition for a limited number of outputs.

times) than L, while if the number of multiplications is to be minimized, P should be chosen about three (2.77) times the size of L. Fig. 9 shows the total number of operations to compute a length 512 DFT for different values of P, and it can be seen that the optimum value of P changes with the number of computed output points. For the filtering approach the numbers become

$$P_{\text{TOT-MIN-TD-FILT}} = \left\lfloor \frac{\sqrt{\left(\frac{N}{\log_e 2}\right)^2 + 6LN + 8N} - \left(\frac{N}{\log_e 2}\right)}{2} \right\rfloor \quad (36)$$

to minimize the total number of operations, and

$$P_{\text{MUL-MIN-TD-FILT}} = \lfloor 2(L+2)\log_e 2 \rfloor \quad (37)$$

to minimize the number of multiplications.

If $X(0)$ is included in the set of needed output points, a further reduction in the number of operations can be achieved by introducing a special butterfly for computing $X(0)$. The number of operations becomes

$$\#\text{MUL}_{\text{TD-FILT-2BF}} = N\log_2 P - 3N + 2(L+1)\frac{N}{P} + 2L - 2 \quad (38)$$

multiplications and

$$\#\text{ADD}_{\text{TD-FILT-2BF}} = 3N\log_2 P - 3N + 2(2L+1)\frac{N}{P} - 2L \quad (39)$$

additions. Fig. 10 compares the four different transform decomposition algorithms discussed in this paper. The split-radix FFT takes a total of 15 368 operations to compute a length 512 DFT, and hence it can be seen that the transform decomposition method is more efficient than computing all the DFT points using the split-radix FFT up to more than 128 output points. Fig. 11 shows the useful region of this method as a function of N, and it can be seen that transform decomposition is more efficient than regular (split-radix)FFT's for any number of outputs while the filtering approach is only efficient up to slightly more than a quarter of the output points. However, this is the region where special methods are most attractive (i.e., offers large savings over regular FFT's) and hence the filtering should nearly always be used. Fig. 12 shows a graph of the total number of operations required by the different methods to compute various numbers of DFT output points of a length 512 DFT. It is evident that the transform decomposition method is far better than the pruning for any L. When L gets comparable to N, the savings over the full length FFT becomes minimal, and for $L \geq N/2$ it is more efficient to compute all N output points

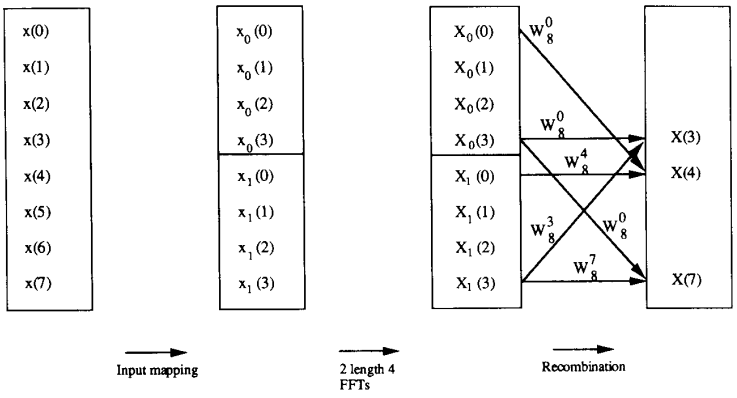


Fig. 8. Block diagram for generalized transform decomposition to compute 3 output points from a length 8 transform.

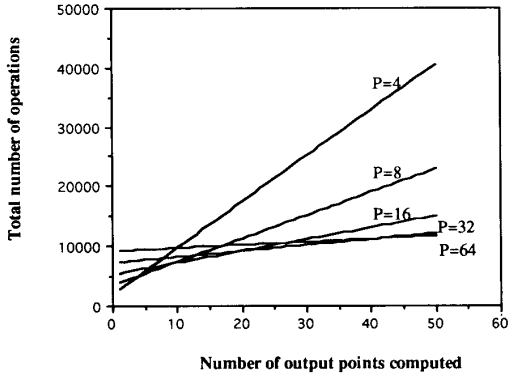


Fig. 9. The total number of operations to compute a subset of output points of a length 512 DFT for different transform lengths P.

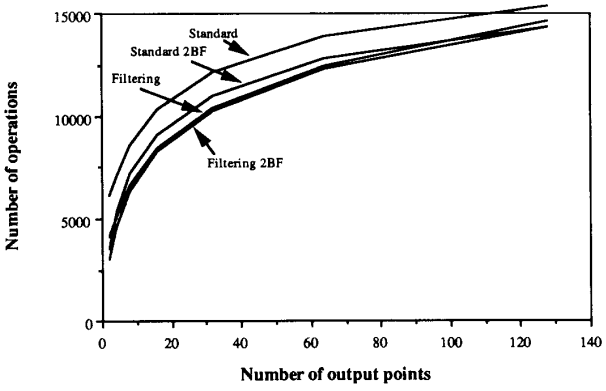


Fig. 10. Comparison of the total number of operations of the four variations of transform decomposition discussed in the section for a DFT length of 512.

using a split-radix FFT. It can also be seen that the filtering approach did not lower the number of operations very much, but since there are no drawbacks, except a few more lines of code, it should generally be used.

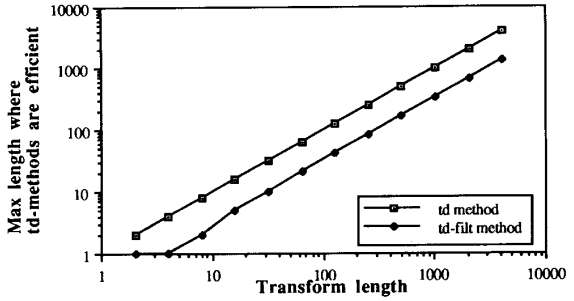


Fig. 11. Crossover point where the split-radix FFT and transform decomposition take the same number of operations.

C. Transform Decomposition for Real Input Data

The method can readily be extended to handle real input data [15]. If the input data is real, the input to the FFT is real, and the length P FFT can be replaced by any length P real input FFT [17]. We will use the real-valued split-radix FFT, which requires

$$\#\mathrm{MUL}_{\mathrm{RFFT}}(N) = \frac{N}{2}\log_2 N - \frac{3N}{2} + 2 \qquad (40)$$

multiplications and

$$\#\mathrm{ADD}_{\mathrm{RFFT}}(N) = \frac{3N}{2}\log_2 N - \frac{5N}{2} + 2 \qquad (41)$$

additions, which is about half that of the complex input split-radix FFT. It is generally not possible to reduce the recombination in (26) so the transform decomposition method requires

$$\#\mathrm{MUL}_{\mathrm{TD\text{-}FILT\text{-}REAL}} = \frac{N}{2}\log_2 P + \frac{N}{2} + 2\frac{N}{P} + 2P \qquad (42)$$

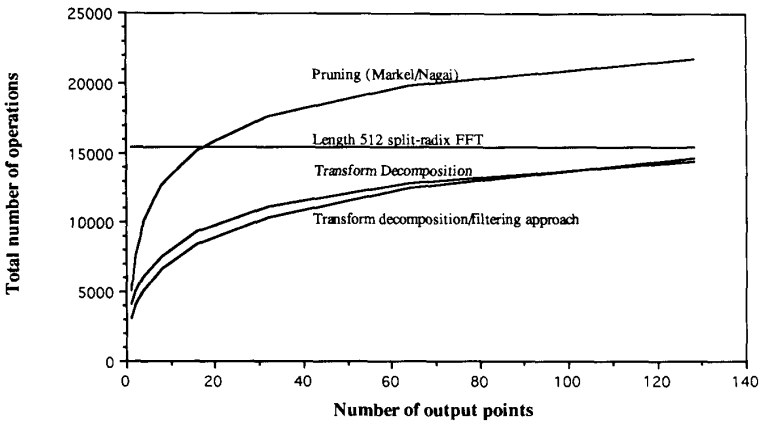


Fig. 12. Performance of different methods for computing a subset of output points of a length 512 DFT.

real multiplications and

$$\#\text{ADD}_{\text{TD-FILT-REAL}} = \frac{3N}{2}\log_2 P + \frac{3N}{2} + 2\frac{N}{P} \quad (43)$$

real additions with real input data.

III. A Subset of Input Points

One way of computing a very high resolution spectrum is to pad the input sequence with a large number of zeros before computing the DFT. It is common to ignore the redundancy in the input sequence and just use a standard FFT, but if the number of actual input values is small compared to the DFT length, this is a very inefficient way of doing things. This section discusses ways of taking advantage of the input redundancy to reduce the number of operations below that of the regular FFT. This problem is very similar, and in some sense dual, to the problem of computing only a subset of DFT output points discussed in the previous section. Goertzel's algorithm [7] can easily be modified to handle this case, but, as for a subset of output points, it is very inefficient. Given that there are L nonzero input points, each output point takes L complex multiplications and $L - 1$ complex additions. Since N output points are needed, the number of operations are $O(NL)$, which approaches N^2 as L approaches N. A more efficient way to use the redundancy is the pruning and transform decomposition methods described in the next sections.

A. Pruning

The pruning method was first developed by Markel [9] for computing a subset of output points of a DFT, see Section I-A. It is obtained by pruning a radix-2 decimation-in-frequency FFT, as is shown in Fig. 13, and it can be found to require

$$\#\text{MUL}_{\text{PRUNE}} = 2N\lfloor\log_2 L\rfloor + 2N - 4L + \frac{2NL}{2^{\lfloor\log_2 L\rfloor}} \quad (44)$$

multiplications and

$$\#\text{ADD}_{\text{PRUNE}} = 3N\lfloor\log_2 L\rfloor + 3N - 6L + \frac{3NL}{2^{\lfloor\log_2 L\rfloor}} \quad (45)$$

additions to compute N output points with only L nonzero input points, where the function $\lfloor\ \rfloor$ returns the integer part of its argument. Comparing these number of operations to the ones in (2) and (3) for the case of a subset of output points, it can be seen they are the same. Similarly, if the L input values are restricted to $0, 1, \cdots, L-1$ a 5 butterfly version can be developed, which will require

$$\#\text{MUL}_{\text{PRUNE-5BF}} = 2N\lfloor\log_2 L\rfloor - 4L - 2N + 4 + \frac{2NL}{2^{\lfloor\log_2 L\rfloor}} \quad (46)$$

real multiplications and

$$\#\text{ADD}_{\text{PRUNE-5BF}} = 3N\lfloor\log_2 L\rfloor - 2L - 3N + 2 + \frac{3NL}{2^{\lfloor\log_2 L\rfloor}} \quad (47)$$

real additions to compute N output points with only L nonzero input points. By comparing the complexities to those in (6) and (8), it can be seen that the 5 butterfly algorithm for a subset of outputs requires more additions than the 5 butterfly algorithm for a subset of inputs above. The reason for this is that the one butterfly versions only use full butterflies, which requires the same number of operations for either a subset of inputs or outputs, while the 5 butterfly versions use half butterflies as well, and the half butterflies for a subset of inputs, require fewer operations than the one for a subset of outputs. This fact can be seen even clearer when looking at Skinner's [14] algorithm shown in Fig. 14, which further restricts L to be a power of two. That algorithm requires

$$\#\text{MUL}_{\text{SKINNER}} = 2N\log_2 L \quad (48)$$

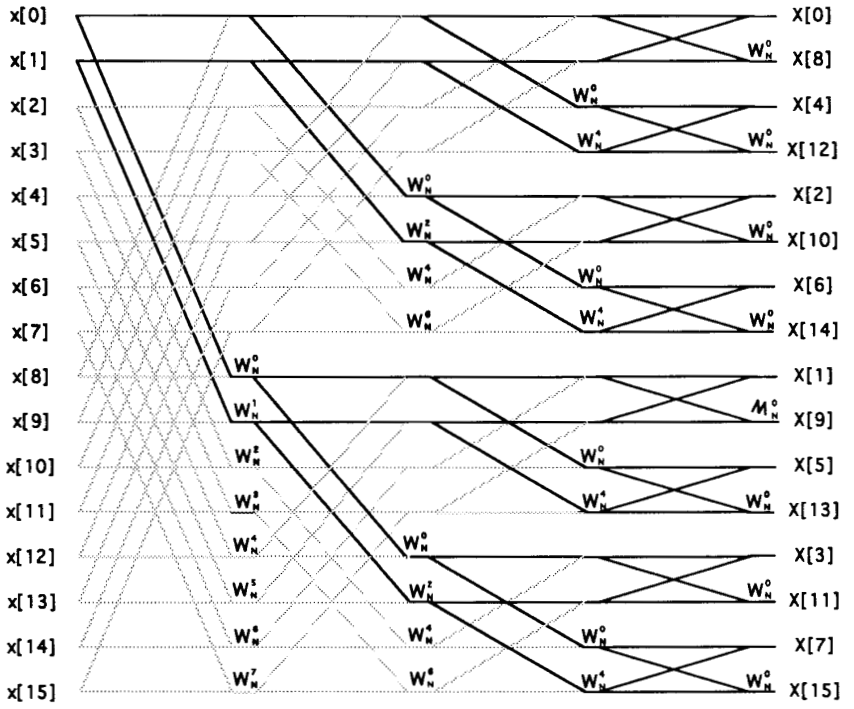


Fig. 13. Length 16 pruned FFT with a subset of nonzero input data.

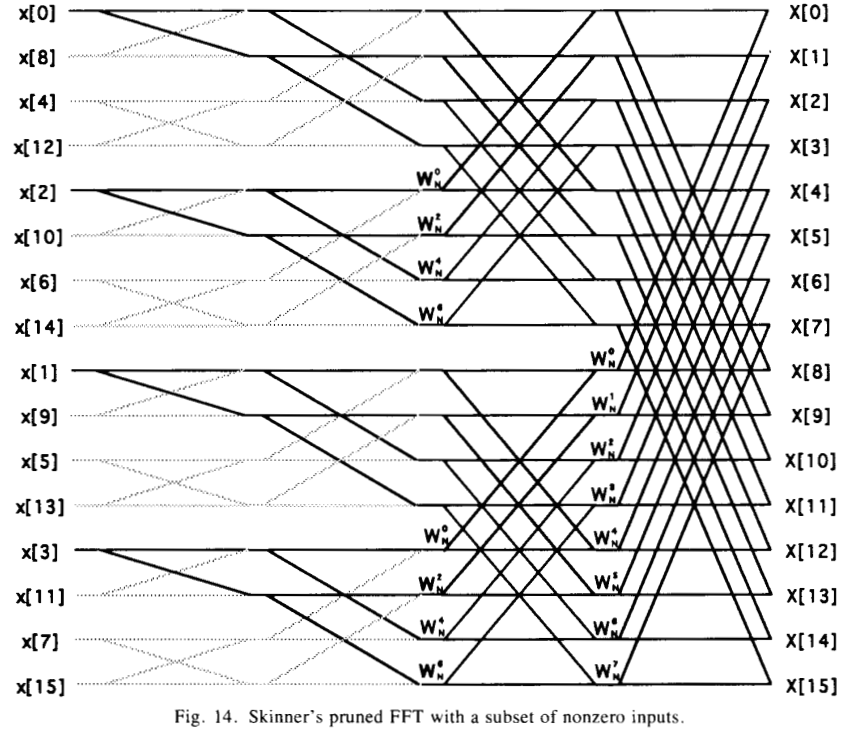


Fig. 14. Skinner's pruned FFT with a subset of nonzero inputs.

real multiplications and

$$\#\text{ADD}_{\text{SKINNER}} = 3N \log_2 L \tag{49}$$

real additions to compute N output points with only L nonzero input points. In Fig. 2 the last two stages contain only half butterflies, which consist of a complex addition. In Fig. 13, however, the first two stages contain half butterflies, which each only is a "copy" operation rather than

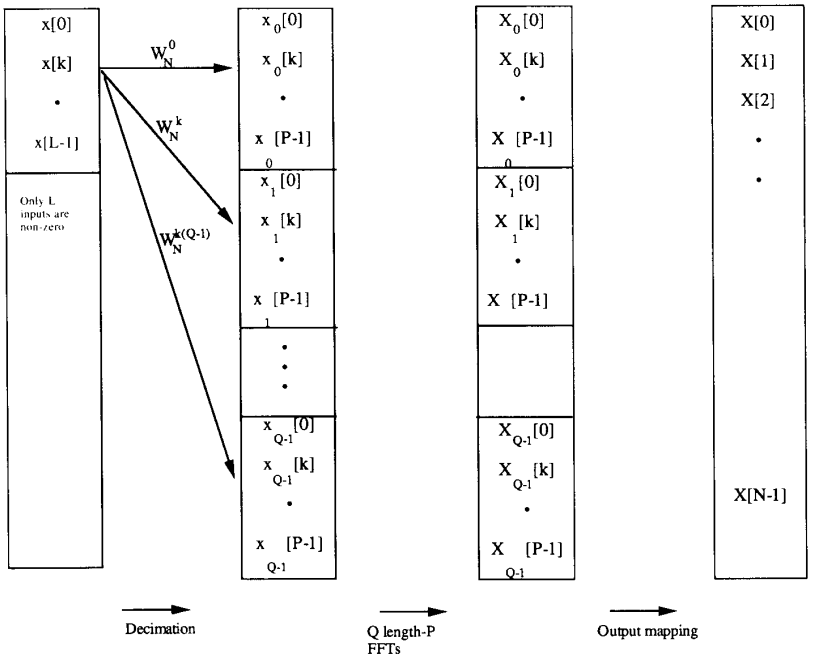


Fig. 15. Block diagram for transform decomposition.

an addition. Hence the algorithm in Fig. 2 requires more additions, but for each extra addition in Fig. 2 there is a data transfer, which will use nearly as much CPU time in Fig. 13. However, just as in the case of a subset of output points, Skinner's algorithm is the best of the pruning algorithms, if the restrictions put on the inputs are not too prohibitive.

B. Transform Decomposition

A new method for computing a DFT of a sequence that only has "few" nonzero entries is developed, which is similar to transform decomposition for computing a subset of DFT output points. Let the DFT be defined

$$X(k) = \sum_{n=0}^{N-1} x(n)W_N^{nk} k = 0, 1, \cdots, N-1 \quad (50)$$

as in (1). Assume that only L input points are nonzero, and that there exists a P such that P divides N and define $Q = N/P$. Now define the variable substitution

$$n = Pn_1 + n_2 \qquad \begin{matrix} n_1 = 0, 1, \cdots, Q-1 \\ n_2 = 0, 1, \cdots, P-1 \end{matrix} \quad (51)$$

and

$$k = k_1 + Qk_2 \qquad \begin{matrix} k_1 = 0, 1, \cdots, Q-1 \\ k_2 = 0, 1, \cdots, P-1 \end{matrix} \quad (52)$$

such that the DFT in (50) can be written

$$X(k_1 + Qk_2) = \sum_{n_2=0}^{P-1} \sum_{n_1=0}^{Q-1} x(Pn_1 + n_2)W_N^{(Pn_1+n_2)(k_1+Qk_2)} \quad (53)$$

$$= \sum_{n_2=0}^{P-1} \left[\sum_{n_1=0}^{Q-1} x(Pn_1 + n_2)W_N^{(Pn_1+n_2)k_1} \right] \cdot W_P^{n_2k_2}. \quad (54)$$

Breaking this up into two equations

$$X_{k_1}(k_2) = \sum_{n_2=0}^{P-1} x_{k_1}(n_2)W_P^{n_2k_2} \qquad k_2 = 0, 1, \cdots, P-1 \quad (55)$$

where

$$x_{k_1}(n_2) = \sum_{n_1=0}^{Q-1} x(Pn_1 + n_2)W_N^{k_1(Pn_1+n_2)} \qquad k_1 = 0, 1, \cdots, Q-1 \quad (56)$$

and

$$X_{k_1}(k_2) = X(k_1 + Qk_2). \quad (57)$$

Equation (55) can, for a given k_1, be recognized as a length P DFT, which can be computed efficiently using an FFT algorithm. As was the case for computing a subset of output points, this is a great advantage of the transform decomposition method over the pruning method, which uses a simple radix-2 FFT. Fig. 15 shows how transform decomposition computes a DFT with only L nonzero input points, and it should be compared to Fig. 4. Equation (56) simplifies when $x(0), x(1), \cdots, x(L-1)$ are the only nonzero inputs and when $L < P$, since only the $n_1 = 0$ terms of (56) remains:

$$x_{k_1}(n_2) = x(n_2)W_N^{k_1n_2} \qquad \begin{matrix} k_1 = 0, 1, \cdots, Q-1 \\ n_2 = 0, 1, \cdots, L=1. \end{matrix} \quad (58)$$

Since a length P FFT is needed for each of the values of k_1 in (55), there are Q of them, and rewriting (58) as

$$x_{k_1}(0) = x(0) \qquad n_2 = 0$$
$$x_{k_1}(n_2) = x(n_2)W_N^{k_1 n_2} \qquad \text{else} \tag{59}$$

the transform decomposition method requires

$$\#\text{MUL}_{\text{TD}} = N \log_2 P - 3N + 4\frac{N}{P} + 4\frac{LN}{P} - 4L \tag{60}$$

multiplications and

$$\#\text{ADD}_{\text{TD}} = 3N \log_2 P - 3N + 4\frac{N}{P} + 2\frac{LN}{P} - 2L \tag{61}$$

additions. Comparing this to (22) and (23) it can be seen that this is the same number of multiplications, but fewer additions, than the case of a subset of output points. This is due to the same reasons that the pruning algorithms for a subset of inputs were more efficient than the pruning algorithms for a subset of outputs; namely that the decimation stage here requires only multiplications, while the recombination stage for the case of a subset of outputs contains both multiplications and additions. Also deselecting the case of $k_1 = 0$, which can be done quite easily by changing the loop indices in the program code, a two butterfly version is obtained, which has the complexity

$$\#\text{MUL}_{\text{TD-2BF}} = N \log_2 P - 3N + 4\frac{LN}{P} - 4L + 4 \tag{62}$$

multiplications and

$$\#\text{ADD}_{\text{TD-2BF}} = 3N \log_2 P - 3N + 2\frac{N}{P} + 2\frac{LN}{P} - 2L + 2 \tag{63}$$

additions. A program achieving these operations counts can be found in Fig. 16. Because these equations assume that the summation in (56) simplifies to the expression in (59), they are only true for $P > L$, which is the interesting range anyway, as can be seen below.

Transform decomposition for a subset of input points is just as flexible as it was for a subset of output points, and Fig. 17 shows how to compute the length 8 DFT with only three nonzero input points, $x(3)$, $x(4)$, $x(7)$ and a transform length of $L = 4$. It should be compared to Fig. 8, which shows the similar case for computing a subset of output points. This is much more versatile than the pruning methods which restrict you to consecutive values and, in the case of Skinner's algorithm, the number of output points has to be a power of two.

To examine the best relationship between P and L, the two expressions for the number of multiplications and additions are added, and the derivative with respect to P is taken. It is then found that to minimize the total number

```
CC=================================================================CC
CC                                                                 CC
CC  Subroutine LITDFT(X,Y,M,R,D,E)                                 CC
CC      A transform-decomposition program when only the first P    CC
CC      input points are non-zero                                  CC
CC      Decimation-in-frequency, cos/sin in second loop            CC
CC      and is computed recursively                                CC
CC                                                                 CC
CC  Input/output                                                   CC
CC      X     Array of real part of input (length >= N)            CC
CC      Y     Array of imaginary part of input (length >= N)       CC
CC      M     Transform length is N=2**M                           CC
CC      R     The number of non-zero input points is P=2**R        CC
CC      D     Array of real part of output (length >= N)           CC
CC      E     Array of imaginary part of output (length >= N)      CC
CC                                                                 CC
CC  Calls:                                                         CC
CC      FFT                                                        CC
CC                                                                 CC
CC  Author:                                                        CC
CC      Henrik Sorensen,  University of Pennsylvania, Oct. 1987    CC
CC                        Arpa address: hvs@pender.ee.upenn.edu    CC
CC  Modified:                                                      CC
CC      Henrik Sorensen,  University of Pennsylvania, Feb. 1988    CC
CC                                                                 CC
CC      This program may be used and distributed freely as         CC
CC      as long as this header is included                         CC
CC                                                                 CC
CC=================================================================CC
      SUBROUTINE LITDFT(X,Y,M,R,D,E)
      REAL X(1),Y(1),D(1),E(1)
      REAL CC1,SS1,CD1,SD1,ARG,T1
      INTEGER P,R,N,M,Q,J,L,K
      N = 2**M
      P = 2**R
      Q=N/P
      DO  60 J=1,Q-1
          L = J*P+1
          X(L) = X(1)
          Y(L) = Y(1)
          ARG = 6.28318530718/N*J
          CC1=COS(ARG)
          SS1=SIN(ARG)
          CD1=CC1
          SD1=SS1
          DO  40 K=2,P
              X(L+K-1) = CC1*X(K) + SS1*Y(K)
              Y(L+K-1) = CC1*Y(K) - SS1*X(K)
C Recursively update trig. functions
              T1 =CC1*CD1-SS1*SD1
              SS1=CC1*SD1+SS1*CD1
              CC1=T1
40        CONTINUE
          CALL FFT(X(L),Y(L),P,R)
          DO  50 K=0,P-1
              D(Q*K+J+1) = X(L+K)
              E(Q*K+J+1) = Y(L+K)
50        CONTINUE
60    CONTINUE
      CALL FFT(X(1),Y(1),P,R)
      DO  70 K=0,P-1
          D(Q*K+1) = X(K+1)
          E(Q*K+1) = Y(K+1)
70    CONTINUE
      RETURN
      END
```

Fig. 16. Program for transform decomposition for a limited number of inputs.

of operations P should be chosen as

$$P_{\text{TOT-MIN-TD}} = \left\lfloor \frac{(3L + 4) \log_e 2}{2} \right\rfloor \tag{64}$$

where $\lfloor\ \rfloor$ indicates "closest" power of two. If instead the lowest possible number of multiplications is required. P should be chosen as

$$P_{\text{MUL-MIN-TD}} = \lfloor 4(L + 1) \log_e 2 \rfloor . \tag{65}$$

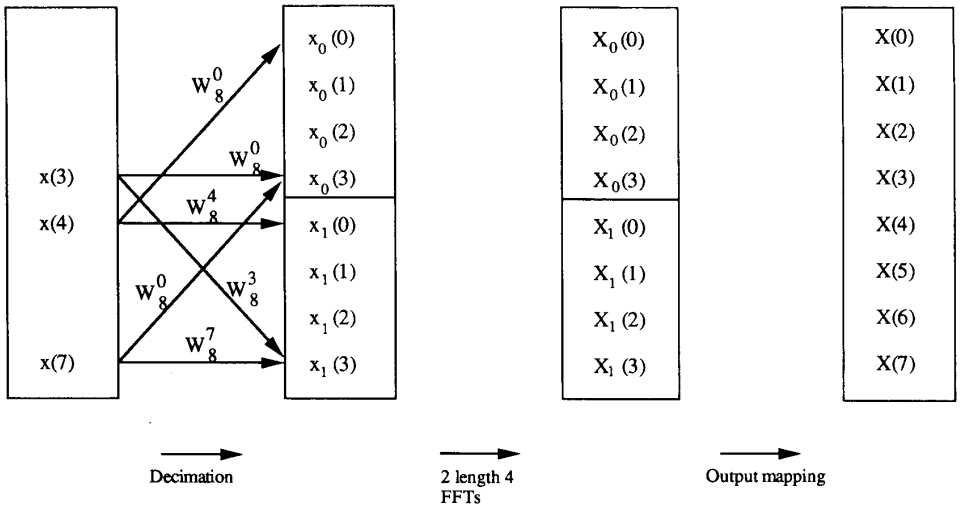


Fig. 17. Generalized transform decomposition with an arbitrary number of input points and any subset.

Similarly, for the 2 butterfly version

$$P_{\text{TOT-MIN-TD-2BF}} = \left\lfloor \frac{(3L + 1)\log_e 2}{2} \right\rfloor \tag{66}$$

and

$$P_{\text{MUL-MIN-TD}} = \lfloor 4L \log_e 2 \rfloor . \tag{67}$$

Notice that since the operation counts these equations are based on are only valid for $P > L$, the same is true for these equations. The optimum P for minimizing the total number of operations and the number of multiplications both satisfy this criterion.

The filtering approach used for the case of a subset of output points applies here only in a limited sense. The summation in (56) can be computed more efficiently using the filtering approach than by direct computation, but for the summation to have more than one term, the transform lengths P must be smaller than the number of outputs L. This is seldom the case and hence the filtering approach is not very useful.

To compare the various methods for taking advantage of zeros in the input sequence of a DFT, Fig. 18 shows the total number of operations to compute three of those algorithms discussed in this section for a length 512 DFT. Shown is also the operation count for the length 512 split-radix FFT, as none of the methods should be used if they require more operations than computing a full DFT using a split-radix FFT. It can be seen that the transform decomposition method is far better than the pruning methods and, opposite to the case of a subset of output points in Fig. 12, the transform decomposition is more efficient than the split-radix for L ranging all the way up to N.

C. Transform Decomposition for Symmetric Input Data

Given that the input data is symmetric (and hence the output must be real) it is easy to see that the input to each

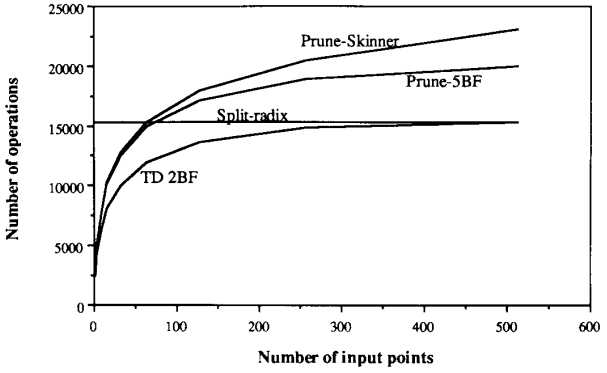


Fig. 18. Comparison of several methods for computing a DFT efficiently with only a subset of input points for a length 512 DFT.

of the FFT must be symmetric, so it produces real output. The symmetric split-radix FFT requires

$$\#\text{MUL}_{\text{SFFT}}(N) = \frac{N}{2}\log_2 N - \frac{3N}{2} + 2 \tag{68}$$

multiplications and

$$\#\text{ADD}_{\text{SFFT}}(N) = \frac{3N}{2}\log_2 N - \frac{5N}{2} + 2 \tag{69}$$

additions, which is about half that of the complex FFT. Since the input to each of the Q DFT's is symmetric, it is only necessary to compute half of the input points, and hence the transform decomposition method can be found to require

$$\#\text{MUL}_{\text{TD-SYMMETRIC}} = \frac{N}{2}\log_2 P + \frac{N}{2} + \frac{2N}{P} - 2P \tag{70}$$

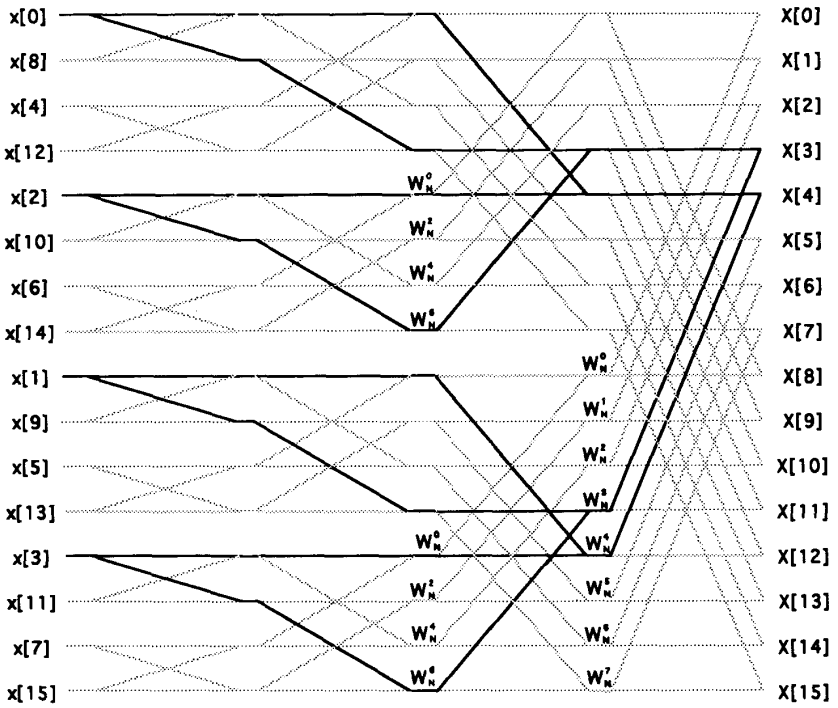


Fig. 19. Markel's pruning on the output with Skinner's pruning on input.

real multiplications and

$$\#\text{ADD}_{\text{TD-SYMMETRIC}} = \frac{3N}{2}\log_2 P - \frac{N}{2} + \frac{2N}{P} - 2P \quad (71)$$

real additions with symmetric input data. Hence the order of the number of operations is halved when the input is symmetric using the transform decomposition method, while it is not possible to lower the operations using pruning.

IV. Both Input and Output Limitations

When both the number of inputs and outputs are less than N, it is possible to achieve algorithms that use fewer operations than can be obtained by using algorithms which take advantage of either a subset of input points or a subset of output points. Looking at the regular pruning schemes in Figs. 1 and 12 for a subset of outputs and inputs, respectively, it can be seen that the subset of output algorithm uses a decimation-in-time algorithm, while the subset of input algorithm uses a decimation-in-frequency algorithm. Looking at Skinner's pruning algorithm, however, it is just the other way around, and hence the way to achieve both input and output pruning is to mix Markel's and Skinner's algorithms. Fig. 19 shows a decimation-in-time length 16 FFT with Skinner's pruning on the input and Markel's on the output. It is, of course, also possible to have Markel's pruning on the input and Skinner's pruning on the output by using a decimation-in-frequency algorithm. This "dual" pruning scheme is described by Sreenivas and Rao [19], [18]. Fig. 20 shows the number of operations required by a decimation-in-time algorithm with Skinner's pruning on the input and Markel's pruning on the output. Since Skinner's algorithm is more efficient than Markel's, the number of operations decrease more by lowering the number of inputs than by lowering the number of inputs in Fig. 20. However, by using the decimation-in-frequency algorithm it is just the other way around, and hence it can be concluded that the decimation-in-time algorithms should be used when the number of inputs are lower that the number of outputs, and the decimation-in-frequency when there are more in-

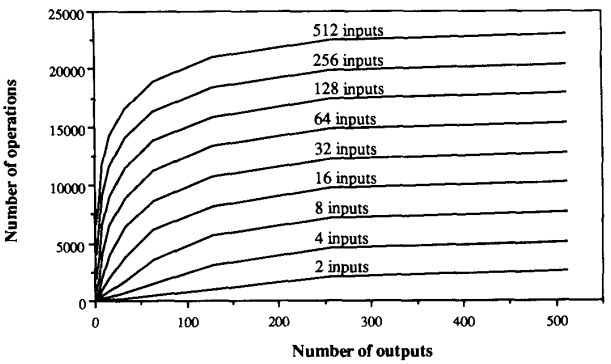


Fig. 20. Number of operations for computing length 512 DFT with a subset of inputs and outputs using a decimation-in-time algorithm.

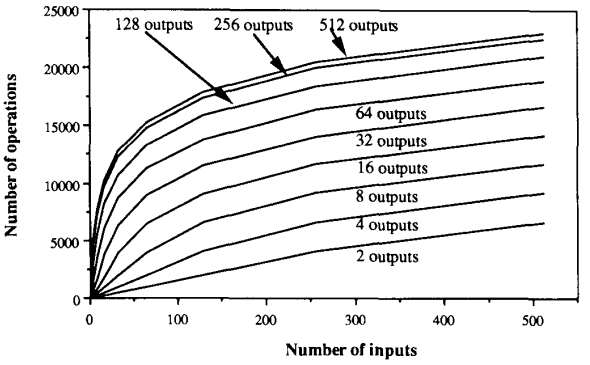


Fig. 21. Number of operations for computing length 512 DFT with a subset of inputs and outputs using a decimation-in-time algorithm.

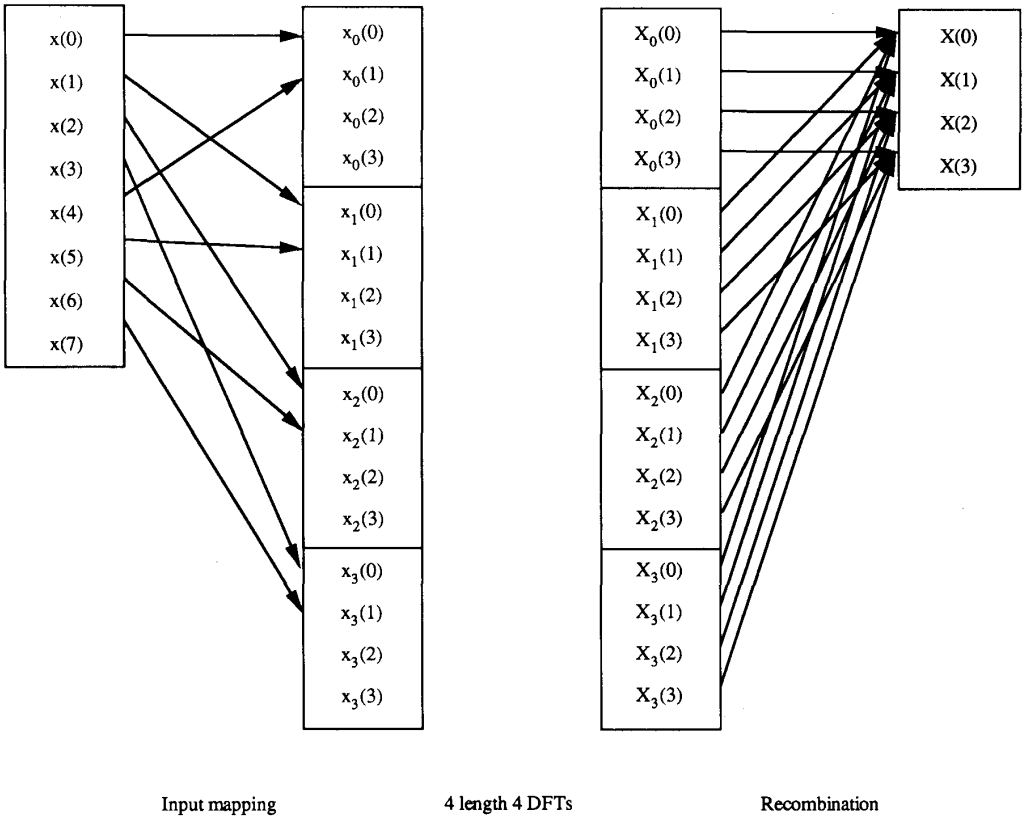


Fig. 22. A length 16 transform decomposition algorithm for computing 4 outputs.

puts than outputs. It can also be seen from Fig. 19, that for a given number of inputs, the number of operations does not decrease very fast as the number of outputs decreases. This point is more clear if Fig. 20 is compared to Fig. 21, which shows the same data as Fig. 20, but the operations here are functions of the number of inputs with the number of outputs as the free parameter.

The transform decomposition method does not seem to be very useful when both the inputs and outputs are limited. It can be found that the majority of the DFT is computed using the decomposition and recombination stages from Figs. 13 and 4. Only a very small portion is done using an FFT algorithm, and hence the effectiveness of the transform decomposition method is lost. Fig. 22 shows a length 16 transform decomposition method for computing outputs. It is here assumed that there are only four nonzero inputs, and it can be seen that the eight inputs are distributed with two in each DFT, and hence the 4 DFT

can be computed using the transform decomposition algorithms for a subset of input points, and each DFT will turn into 2 length 2 DFT's. Notice, however, this method does not depend on the transform length, the number of inputs, or the number of outputs. What determines the structure of the transform is the ratio between the number of inputs and the number of outputs. If the ratio is one, the DFT's are of length 1, which is not very interesting. That means that all the computations are done in the "decimation" and the "recombination" stages of the algorithm, which is not very efficient compared to an FFT algorithm. Hence we will have to conclude the transform decomposition method is less efficient than pruning when both the number of inputs and outputs are limited.

V. Conclusion

Given the fact that only L of the N DFT samples are "needed," there are several ways to perform the computations. One can ignore the fact that only a subset of the output points is needed, and use an FFT to compute all N output points, or one can use Goertzel's [7] or Boncelet's [2] methods to compute individual output points. These three methods require so many operations, even for small L, that they are uninteresting and will not be considered any further.

It is evident that the transform decomposition methods are far better than the pruning for any L. When L gets comparable to N, the savings over the full length FFT becomes minimal, and for $L \geq N/2$ it is more efficient to compute all N output points using a split-radix FFT. It can also be seen that the filtering approach did not lower the number of operations very much, but since there are no drawbacks from using them, except a few more lines of code, they should generally be used. Two other advantages of the transform decomposition methods over pruning are, first, that transform decomposition also gives a significant savings for real input or output data, while it is very difficult, if not impossible, to prune a real input FFT. Second, the computed output points do not have to be sequential as they do in pruning.

Acknowledgment

The authors would like to thank J. Cooley for his useful suggestions.

References

[1] G. D. Bergland, "A radix-eight fast-Fourier transform subroutine for real-valued series," *IEEE Trans. Audio Electroacoust.*, vol. AU-17, no. 2, pp. 138-144, June 1969.

[2] C. G. Boncelet, Jr, "A rearranged DFT algorithm requiring $N^2/6$ multiplications," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-34, no. 6, pp. 1658-1659, Dec. 1986.

[3] C. S. Burrus and T. W. Parks, *DFT/FFT and Convolution Algorithms*. New York: Wiley, 1985.

[4] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series," *Math. Comput.*, vol. 19, no. 90, pp. 297-301, Apr. 1965.

[5] G. C. Danielson and C. Lanczos, "Some improvements in practical Fourier analysis and their application to X-ray scattering from liquids," *J. Franklin Inst.*, vol. 233, nos. 4 and 5, pp. 365-380 and 435-452, Apr. and May 1942.

[6] R. de Wild, "Method for partial spectrum computation," *Proc. Inst. Elec. Eng.*, vol. 134, no. 7, pp. 659-666, Dec. 1987.

[7] G. Goertzel, "An algorithm for the evaluation of finite trigonometric series," *Amer. Math. Monthly*, vol. 65, no. 1, pp. 34-35, Jan. 1958.

[8] D. P. Kolba and T. W. Parks, "A prime factor FFT algorithm using high-speed convolution," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-25, no. 4, pp. 281-294, Aug. 1977.

[9] J. D. Markel, "FFT pruning," *IEEE Trans. Audio Electroacoust.*, vol. AU-19, no. 4, pp. 305-311, Dec. 1971.

[10] K. Nagai, "Pruning the decimation-in-time FFT algorithm with frequency shift," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-34, no. 4, pp. 1008-1010, Aug. 1986.

[11] A. V. Oppenheim and R. W. Schafer, *Digital Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1975.

[12] C. Runge, "Ueber die zerlegung empirisch gegebener periodischer funktionen in sinuswellen," *Z. Math. Phys.*, vol. 48, pp. 443-456, 1903 (in Germany).

[13] R. C. Singleton, "An algorithm for computing the mixed radix fast Fourier transform," *IEEE Trans. Audio Electroacoust.*, vol. AU-17, no. 2, pp. 93-103, June 1969.

[14] D. P. Skinner, "Pruning the decimation in-time FFT algorithm," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-24, no. 2, pp. 193-194, Apr. 1976.

[15] H. V. Sorensen and C. S. Burrus, "A new efficient algorithm for computing a few DFT points," in *Proc. IEEE 1988 Int. Symp. Circuits Syst.*, June 7-9, 1988, pp. 1915-1918.

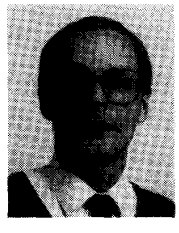
[16] H. V. Sorensen, M. T. Heideman, and C. S. Burrus, "On computing the split-radix FFT," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-34, no. 11, pp. 152-156, Feb. 1986.

[17] H. V. Sorensen, D. L. Jones, M. T. Heideman, and C. S. Burrus, "Real-valued fast Fourier transform algorithms," *IEEE Trans. Acoust., Speech, Signal Processing*. vol. ASSP-35, no. 6, pp. 849-863, June 1987.

[18] T. V. Sreenivas and P. V. S. Rao, "High resolution narrow-band spectra by FFT pruning," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-28, no. 2, pp. 254-257, Apr. 1980.

[19] T. V. Sreenivas and P. V. S. Rao, "FFT algorithm for both input and output pruning," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-27, no. 3, pp. 291-292, June 1979.

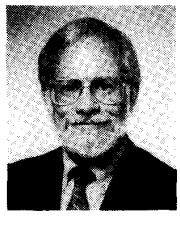
[20] S. Winograd, "On computing the discrete Fourier transform," *Math. Comput.*, vol. 32, no. 141, pp. 175-199. Jan. 1978.



Henrik V. Sorensen (S'84-M'88) was born in Skanderborg, Denmark, on January 17, 1959. He received the B.S. and M.S. degrees in electrical engineering from Aalborg University Center, Aalborg, Denmark, in 1981 and 1983, respectively, and the Ph.D. degree in electrical engineering from Rice University, Houston, TX, in 1988.

He is presently an Assistant Professor in the Department of Electrical Engineering at the University of Pennsylvania. His research interests are in the area of digital signal and image processing.

Dr. Sorensen is a member of Sigma Xi, Eta Kappa Nu, DIF, and the Danish Engineering Society. He is currently an Associate Editor for the IEEE Transactions on Signal Processing.



C. Sidney Burrus (S'55-M'61-SM'75-F'81) was born in Abilene, TX, on October 9, 1934. He received the B.A. B.S.E.E., and M.S. degrees from Rice University, Houston, TX, in 1957, 1958, and 1960, respectively, and the Ph.D. degree from Stanford University, Stanford, CA, in 1965.

From 1960 to 1962 he taught at the Navy Nuclear Power School in New London, CT, and during the summers of 1964 and 1965 he was a Lecturer in Electrical Engineering at Stanford University. In 1965 he joined the faculty at Rice

University where he is now Professor and Chairman of the Electrical and Computer Engineering. From 1972 to 1978 he was Master of Lovett College at Rice University. In 1975–1976 and again in 1979–1980, he was a Guest Professor at the Universitaet Erlangen-Nuernberg, Germany. During the summer of 1984 he was a Visiting Fellow at Trinity College, Cambridge University, Cambridge, England and during the academic year 1989–1990 he was a Visiting Professor at M.I.T., Cambridge, MA.

Dr. Burrus is a member of Tau Beta Pi and Sigma Xi. He received teaching awards from Rice University in 1969, 1974, 1975, 1976, 1980, and 1989, an IEEE S-ASSP Senior Award in 1974, a Senior Alexander von Humboldt Award in 1975, a Senior Fulbright Fellowship in 1979, and the IEEE S-ASSP Technical Achievement Award in 1985. He has coauthored two books in digital signal processing. He is currently on the IEEE Signal Processing Society's Administrative Committee.