loop. It is therefore expected that this implementation will be almost as fast as the fixed size transform program.

Execution times were tested for three PFA implementations.
1) PFAT-The program described here.
2) PFA-The in-place transform with separate unscrambler from [1].
3) PFA2-The in-place, in-order, fixed size transform from [2].

In all three programs, the nested subscripts in the DFT modules were eliminated by equivalencing the elements of the I and IR arrays to integer variables.

Times were measured on a Hewlett-Packard 2108M computer with fast Fortran package using the HP Fortran IV compiler. For comparison, times for a fast radix-8 FFT [3] are also presented. Results are shown in Table I. This program is only slightly slower than a fixed size transform; it is also faster than the use of a separate unscrambling pass since the added data transfers are eliminated.

**SUMMARY**

A revised version of the in-place, in-order prime factor FFT has been described. Speed measurements indicate that this version is faster than the use of a separate unscrambling pass and is close in speed and complexity to a program optimized for a fixed size transform. For variable size transforms, memory requirements are substantially reduced since the separate output array for unscrambling is traded for a much smaller output address array used by the short DFT modules.

**REFERENCES**


**Selection Criteria for Efficient Implementation of FFT Algorithms**

J. D. BLANKEN AND P. L. RUSTAN

Abstract—The number of real operations and memory is presented for three efficient Fortran algorithms which compute the mixed radix discrete Fourier transform (DFT). It is shown that Singleton's mixed radix algorithm (MFFT) is the most flexible and uses the least memory.

Manuscript received January 16, 1981; revised July 29, 1981. The authors are with the Department of Electrical Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH 45433.

**TABLE I**

<table>
<thead>
<tr>
<th>TRANSFORM SIZE</th>
<th>PFAT</th>
<th>PFA</th>
<th>PFA2</th>
<th>FFT842</th>
</tr>
</thead>
<tbody>
<tr>
<td>60</td>
<td>84</td>
<td>89</td>
<td>80</td>
<td></td>
</tr>
<tr>
<td>64</td>
<td></td>
<td></td>
<td></td>
<td>138</td>
</tr>
<tr>
<td>252</td>
<td>460</td>
<td>500</td>
<td>440</td>
<td></td>
</tr>
<tr>
<td>256</td>
<td></td>
<td></td>
<td></td>
<td>680</td>
</tr>
<tr>
<td>504</td>
<td>990</td>
<td>1050</td>
<td>950</td>
<td></td>
</tr>
<tr>
<td>512</td>
<td></td>
<td></td>
<td></td>
<td>1460</td>
</tr>
</tbody>
</table>

while the Winograd Fourier transform algorithm (WFTA) and Kolba-Parks prime factor algorithm (PFA) are the most efficient.

**INTRODUCTION**

This correspondence presents the general expressions for the number of real multiplications, additions, and memory arrays for the WFTA, PFA, and MFFT. These results are plotted and tabulated for all permissible WFTA and PFA sequence lengths and compared to MFFT. Based on the assumptions that first, the algorithms are not optimized to match any particular machine architecture, and second, that the number of real operations is the best first-order approximation to an algorithm's efficiency, the optimum Fortran mixed radix DFT may be selected given sequence length N, computer speed (adds and multiplies), and computer memory.

**NUMBER OF REAL OPERATIONS AND MEMORY**

Winograd's short DFT algorithms were implemented in Fortran [1] and require the real operations count shown in Table I, where \( N_i \) is a factor of \( N \), and \( M_i \) and \( A_i \) are the number of real multiplications and additions, respectively, for the \( N_i \) factor. The data from Table I was used by Silverman [2] for the number of real operations as a function of \( N \) for the WFTA

\[
NMULT = 2 \prod_{r=1}^{R} M_r 
\]

(1)

\[
NADDS = 2 \sum_{r=1}^{R} \left( \prod_{i=1}^{N_i} M_i \right) \left( \prod_{j=r+1}^{R} M_j \right)
\]

(2)

where

\[
N = N_1 * N_2 * \cdots * N_{R-1} * N_R.
\]

The memory array requirements for the WFTA were determined from the Fortran program listing [1]. There are four arrays of length \( N \), three arrays of length NMULT, and nine miscellaneous arrays which require 88 memory locations for a total of

memory array = 4 * \( N \) + 3 * NMULT + 88. (3)

An alternative to the WFTA nested structure was developed by Kolba and Parks [3]. The PFA converts short DFT's to circular convolution but instead of "nesting" the short DFT's, the PFA uses the "conventional" DFT decomposition of the \( N \) length sequence. The PFA was slightly modified [4] to provide the real operations counts for the PFA short DFT's shown in Table II.

U.S. Government work not protected by U.S. copyright
The general expression for real operations as a function of the sequence length \( N \) was derived from Kolba and Parks [3] as

\[
\text{NMULT} = 2 \sum_{r=1}^{\infty} \left( \prod_{j=1}^{N} N_j \right) M_r \left( \prod_{j=r+1}^{N} N_j \right)
\]

(4)

\[
\text{NADDS} = 2 \sum_{r=1}^{\infty} \left( \prod_{j=1}^{N} N_j \right) A_r \left( \prod_{j=r+1}^{N} N_j \right)
\]

(5)

where \( M_r \) and \( A_r \) are the number of real multiplications and additions, respectively, for the factors of \( N_j \) shown in Table II.

The PFA memory requirements were determined from the Fortran program listing [5]. There are four length \( N \) arrays required by the PFA which gives

memory array \( = 4 \ast N \).

(6)

The MFFT computes the DFT of any positive sequence length \( N \); however, it is most efficient when \( N \) is highly factorable by 2, 3, 4, or 5 [5]. There are special sections to handle these factors as well as a general odd prime number section.

The number of real operations required for the MFFT was derived from the number of complex twiddle (rotation) factors, butterflies (short DFT’s), and the trigonometric functions used to compute the FFT. The MFFT requires that \( N \) be factored as

\[
N = 2^m 3^n 5^p_1 7^p_2 \cdots p_k^m_k
\]

(7)

which reflects the special and general transform sections. Using the notation in (7), a general expression for real operations was determined [6] to be

\[
\text{real mult} = 2rN + 4sN + 3tN + 32uN/5 + \sum_{i=1}^{M} (2p_i - 1) + (m_1) N(p_i - 1)^2 p_i + 4(m_1) N(p_i - 1)/p_i - 4(N - 1) + \text{KMULT}
\]

(8)

\[
\text{real adds} = 3rN + 18sN/3 + 11tN/2 + 8uN + \sum_{i=1}^{M} ((p_i - 1) + 7N(m_1)(p_i - 1)/p_i + (m_1) N(p_i - 1)^2/p_i - 2(N - 1) + \text{KADD}
\]

(9)

where KMULT and KADD are the real operations associated with computing the trigonometric functions [6].

The memory requirements for MFFT are described by Singleton [5]. From his description the general expression for array storage becomes

memory arrays \( = 2 \ast N + 4 \ast \text{MAXPF} + \text{MAX}(K - 1, M + 1) \)

(10)

where MAXPF is the maximum prime factor of \( N \), \( K \) is the product of square free factors, and \( M \) is the maximum number of prime factors.

**Comparison Results of Efficient DFT’s**

**Real Operations:** The MFFT, WFTA, and PFA real operations counts are shown in Figs. 1 and 2. The MFFT results were generated for \( N \) factorable by 2, 3, 4, and 5. The results for WFTA and PFA were generated for all 59 possible sequence lengths which the algorithms can transform. Fig. 1 demonstrates the relative efficiency of these algorithms to the fixed radix-2 complex transform multiplications count of \( 2N \log_2 N \). The WFTA and PFA offer significant improvements in number of real multiplications. Fig. 2 shows that the real addition levels for WFTA and PFA are maintained at the \( 3N \log_2 N \) radix-2 FFT level. Both figures demonstrate a savings in real operations by WFTA and PFA over the MFFT.

The tradeoffs between WFTA and PFA are also seen in Figs. 1 and 2. In most cases the WFTA requires fewer multiplications but more additions than PFA. The selection of the most efficient algorithm then becomes dependent on machine speed of real addition versus real multiplication.

**Memory:** The memory array locations required by the algorithms are presented in Fig. 3 demonstrating MFFT uses the least memory. These results show the efficient data reordering technique of the MFFT which performs the reordering with little additional memory relative to the sequence length. The WFTA and PFA base their data reordering on the Chinese Remainder Theorem and require an additional two length \( N \) arrays for storing the intermediate results. The WFTA uses even more memory than the PFA because of its structure which “nests” multiplications inside all the additions. The
number of real operations and memory arrays are tabulated in Table III.

**Flexibility:** WFTA and PFA are less flexible than MFPT because only short DFT modules for \( N = 2, 3, 4, 5, 7, 8, 9 \) and 16 have been implemented. Other short DFT modules exist [4], but have not been incorporated into mixed radix algorithms.

**Summary**

The general expressions for real operations and memory are given for WFTA, PFA, and MFPT. These expressions are plotted and tabulated so an optimum DFT can be selected based on real operations and memory. Selecting the optimum DFT based on minimizing real operations assumes that none of the algorithms has been optimized to a particular machine architecture.

**References**


**Design of Two-Dimensional Recursive Digital Filters with Coefficients of Finite Word Length**

GIOVANNI L. SICURANZA

Abstract—An approach for the design of two-dimensional (2-D) recursive digital filters with coefficients of finite word length is described. The method is derived from a "continuous" optimization procedure which appears to be particularly suitable to meet both amplitude and phase specifications in the frequency domain. The proposed algorithm finds good solutions in a comparatively short computer time.

**I. Introduction**

Computer-aided techniques for the design of recursive digital filters are of increasing interest for many applications involving two-dimensional (2-D) signal processing, such as image processing, geophysical, and biomedical data analysis. Many reports in the literature concern the analysis of errors resulting from finite word length in the 1- or 2-D case. However, examples of design of 2-D digital filters with integer coefficients are very rare, even though, in theory, most of the methods currently used in the 1-D case could be exploited.

The form of this note is to demonstrate the possibility of designing 2-D digital recursive filters with coefficients of finite word length, and to solve the related integer optimization problem with low computation time. The method we propose is based on the evaluation of a limited set of constrained minima for the chosen error function, moving from the rounded values of the coefficients found in the "continuous" case. Therefore, the method may not find the optimal finite word length solution, although usually a good suboptimal solution is obtained with a reduced number of iterations.

Manuscript received June 11, 1980; revised October 27, 1980 and March 2, 1981. This work was supported by the Italian National Research Council (CNR). The author is with the Istituto di Elettrotecnica ed Elettronica, Università di Trieste, Trieste, Italy.