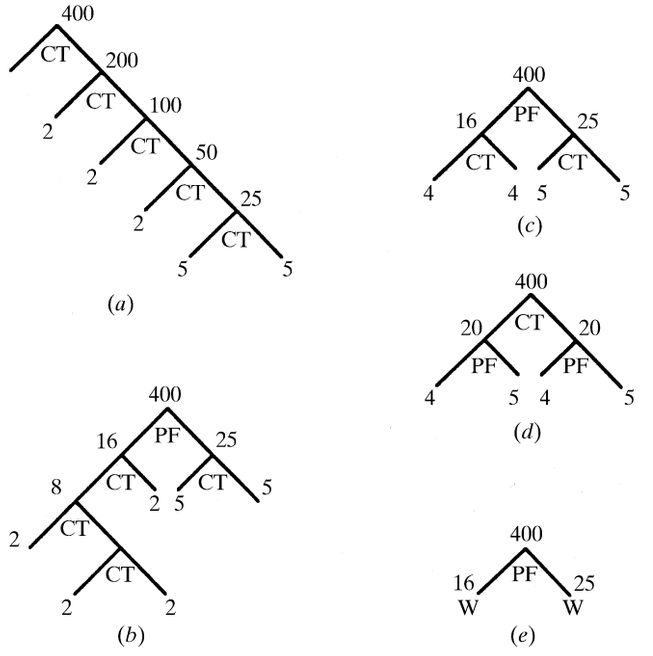

Fig. 1.3.3.1. A few global algorithms for computing a 400-point DFT. CT: Cooley–Tukey factorization. PF: prime factor (or Good) factorization. W: Winograd algorithm.

The simplest DFT may then be carried out into a global algorithm in many different ways. The diagrams in Fig. 1.3.3.1 illustrate a few of the options available to compute a 400-point DFT. They may differ greatly in their arithmetic operation counts.

1.3.3.3.3.2. *Computer architecture considerations*

To obtain a truly useful measure of the computational complexity of a DFT algorithm, its arithmetic operation count must be tempered by computer architecture considerations. Three main types of trade-offs must be borne in mind:

(i) reductions in floating-point (f.p.) arithmetic count are obtained by reindexing, hence at the cost of an increase in integer arithmetic on addresses, although some shortcuts may be found (Uhrich, 1969; Burrus & Eschenbacher, 1981);

(ii) reduction in the f.p. multiplication count usually leads to a large increase in the f.p. addition count (Morris, 1978);

(iii) nesting can increase execution speed, but causes a loss of modularity and hence complicates program development (Silverman, 1977; Kolba & Parks, 1977).

Many of the mathematical developments above took place in the context of single-processor serial computers, where f.p. addition is substantially cheaper than f.p. multiplication but where integer address arithmetic has to compete with f.p. arithmetic for processor cycles. As a result, the alternatives to the Cooley–Tukey algorithm hardly ever led to particularly favourable trade-offs, thus creating the impression that there was little to gain by switching to more exotic algorithms.

The advent of new machine architectures with vector and/or parallel processing features has greatly altered this picture (Pease, 1968; Korn & Lambiotte, 1979; Fornberg, 1981; Swartzrauber, 1984):

(i) *pipelining* equalizes the cost of f.p. addition and f.p. multiplication, and the ideal 'blend' of the two types of operations depends solely on the number of adder and multiplier units available in each machine;

(ii) integer address arithmetic is delegated to specialized arithmetic and logical units (ALUs) operating concurrently with the f.p. units, so that complex reindexing schemes may be used without loss of overall efficiency.

Another major consideration is that of data flow [see *e.g.* Nawab & McClellan (1979)]. Serial machines only have few registers and few paths connecting them, and allow little or no overlap between computation and data movement. New architectures, on the other hand, comprise banks of vector registers (or 'cache memory') besides the usual internal registers, and dedicated ALUs can service data transfers between several of them simultaneously and concurrently with computation.

In this new context, the devices described in Sections 1.3.3.2 and 1.3.3.3 for altering the balance between the various types of arithmetic operations, and reshaping the data flow during the computation, are invaluable. The field of machine-dependent DFT algorithm design is thriving on them [see *e.g.* Temperton (1983*a*,*b*,*c*, 1985); Agarwal & Cooley (1986, 1987)].

1.3.3.3.3.3. *The Johnson–Burrus family of algorithms*

In order to explore systematically all possible algorithms for carrying out a given DFT computation, and to pick the one best suited to a given machine, attempts have been made to develop:

(i) a high-level notation of describing all the ingredients of a DFT computation, including data permutation and data flow;

(ii) a formal calculus capable of operating on these descriptions so as to represent all possible reorganizations of the computation;

(iii) an automatic procedure for evaluating the performance of a given algorithm on a specific architecture.

Task (i) can be accomplished by systematic use of a tensor product notation to represent the various stages into which the DFT can be factored (reindexing, small transforms on subsets of indices, twiddle factors, digit-reversal permutations).

Task (ii) may for instance use the Winograd **CBA** normal form for each small transform, then apply the rules governing the rearrangement of tensor product $\otimes$ and ordinary product $\times$ operations on matrices. The matching of these rearrangements to the architecture of a vector and/or parallel computer can be formalized algebraically [see *e.g.* Chapter 2 of Tolimieri *et al.* (1989)].

Task (iii) is a complex search which requires techniques such as dynamic programming (Bellman, 1958).

Johnson & Burrus (1983) have proposed and tested such a scheme to identify the optimal trade-offs between prime factor nesting and Winograd nesting of small Winograd transforms. In step (ii), they further decomposed the pre-addition matrix **A** and post-addition matrix **C** into several factors, so that the number of design options available becomes very large: the N-point DFT when N has four factors can be calculated in over 10^{12} distinct ways.

This large family of nested algorithms contains the prime factor algorithm and the Winograd algorithms as particular cases, but usually achieves greater efficiency than either by reducing the f.p. multiplication count while keeping the number of f.p. additions small.

There is little doubt that this systematic approach will be extended so as to incorporate all available methods of restructuring the DFT.

1.3.4. Crystallographic applications of Fourier transforms

1.3.4.1. *Introduction*

The central role of the Fourier transformation in X-ray crystallography is a consequence of the kinematic approximation used in the description of the scattering of X-rays by a distribution of electrons (Bragg, 1915; Duane, 1925; Havighurst, 1925*a*,*b*; Zachariasen, 1945; James, 1948*a*, Chapters 1 and 2; Lipson & Cochran, 1953, Chapter 1; Bragg, 1975).

references