

# Unordered Parallel Distance-1 and Distance-2 FFT Algorithms of Radix 2 and (4-2)

Mahn-Ling Woo

Rosemary A. Renaut

Department of Mathematics, Arizona State University

Keywords: Complexity analysis, parallel FFT algorithms, radix-2 FFT, mixed-radix (4-2) FFT

## Abstract

Algorithms for unordered Parallel Fast Fourier transform (PFFT) pairs with radices 2 and mixed radix (4-2) for distributed memory machines are presented. Distributed memory versions using distance one and distance two communications are derived. Theoretical estimates of the computation costs in each case demonstrate that the higher-radix FFT is more efficient for parallel implementation. Furthermore, distance-two communication strategies can minimize communication cost when certain architecture dependent parameters are satisfied.

**Introduction.** In this paper, power-of-two (PO2) Fast Fourier transforms are considered for implementation on the power-of-two topology hypercube. PO2 FFTs can be classified into two groups: unordered and ordered. The unordered PO2 PFFT algorithms produce an output data sequence which is the bit-reverse of the input data sequence. The ordered PO2 PFFT algorithms generate identical input and output data sequences. In this paper we focus on PO2 unordered PFFT algorithms, and, in particular, unordered PFFT pairs consisting of a forward and inverse transform. It has already been demonstrated by Woo and Renaut [6] that the most efficient radix-2 PFFT algorithms minimize the number of communications, the communication distance, and the data packet size.

It is known that, in scalar mode, radix-2 FFT algorithms require more computation than radix-4 and mixed-radix (4-2) FFT algorithms. Is this still the case in parallel mode? Furthermore, as commonly assumed, is distance two communication required? Here we show that indeed higher radix has the potential to be more efficient and further can be implemented in distance one.

The definition of scalar FFT can be found in many text books and papers such as [1, 3]. Parallel implementations are designed with the aid of the sequence to processor map which is explained in the following section. Distance-1 algorithms and their time complexities are explained and compared in the later sections. Conclusions follow at the end.

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.

© 1994 ACM 089791-647-6/ 94/ 0003 \$3.50

**Design of PO2 PFFT Algorithms.** The distribution of the data points among the processors plays a very important role in reducing the number of communications for the FFT. Here an interleaved or wraparound data assignment is used. For this mapping, the data ids in each processor are defined by data id. =  $me + P \cdot j$ ,  $j = 0, \dots, 2^{n-d} - 1$ , where  $P = 2^d$  is the number of processors and  $me$  is the processor id. This is most easily implemented via the interleaved sequence-to-processor map as introduced by Woo and Renaut [4, 2]:

$$x_m = (i_d, \dots, i_{n-1} \mid i_0, \dots, i_{d-1}). \quad (1)$$

Here the element id  $m = i_{n-1} \dots i_0$  (binary) and the partition  $\mid$  has been introduced to separate the address on the left from the processor number on the right. With this notation the element  $x_m$  has address  $i_{n-1} \dots i_d$  in processor  $i_{d-1} \dots i_0$ .

To implement the PFFT, we use index-digit permutations of  $x_m$  as described by the  $r$ -element  $i$ -cycle and partial-exchange  $i$ -cycles, [3] and [4, 5, 7], respectively.

**Definition.** An  $r$ -element  $i$ -cycle is an index-digit permutation of  $x_m$  in which the pivot group is exchanged with any group of  $r$  digits, either in the address or the processor number. Here, the pivot is a group of  $r$  digits.

**Definition.** An  $r$ -element partial-exchange  $i$ -cycle is an index-digit permutation of  $x_m$  in which the pivot is exchanged only with any group of  $r$  digits in the processor number position. Here, the pivot is a group of  $r$  digits.

All the  $i$ -cycles satisfy the following properties:

- (1) The communication distance is equal to the number of different bits in the processor id (in binary form).
- (2) The communication length is equal to  $2^{n-d-v}$  where  $v$  is the number of different bits in address id (binary form).
- (3) Any element in the address position can be the pivot.
- (4) Any exchange in address only means that the data point sequence in each processor must be rearranged and no interprocessor communication is involved.

Note that the distance-one and distance-two communication strategies presented here use  $r = 1$  and  $r = 2$  in the above definitions.

**Distance-1 PO2 Parallel FFT Pairs. Algorithms.** The algorithm for the radix-2 PFFT pair which assigns data in wraparound order was first introduced by Woo [4, 2]. It uses  $d$  distance-1 communications with packets of size  $2^{n-d-1}$  data points and, as such, minimizes communication

cost as discussed briefly in the introduction. Since the radix-4 PFFT is just the mixed-radix (4-2) PFFT but with  $N$  a power of 4, only the mixed-radix (4-2) FFTs are discussed here. In these algorithms the right-most digit in the address position is chosen and the algorithms separate into just three cases:  $n/2 > d$ ,  $n/2 \leq d$  and even ( $\lceil n/2 \rceil$ ) and  $n/2 \leq d$  and odd ( $\lceil n/2 \rceil$ ). Examples of these algorithms are given in Table I.

In the example, ( $idx$ ) indicates the  $idx$ th stage is calculated by partial-exchange  $i$ -cycle. No communication is required here. If '\*' is added, then it indicates the distance-1 communication of length  $2^{n-d-1}$  is performed. 'a' is added for the exchange of pivot without communication and the 'b' is for the exchange of pivot with distance-1 communication.

TABLE I.  
Unordered distance-1 mixed-radix  
PFFT algorithms ( $n = 9$ ,  $d = 3$ ).

| (a) forward   | (b) inverse   |
|---|---|
| $x(i_3 i_4 i_5 i_6 i_7 i_8   i_0 i_1 i_2)$          | $x(k_5 k_4 k_3 k_7 k_6 k_8   k_2 k_1 k_0)$          |
| $X^{(0)}(i_3 i_4 i_5 i_6 k_7 k_8   i_0 i_1 i_2)$    | $X^{(0)}(k_5 k_4 k_3 i_7 k_6 i_8   k_2 k_1 k_0)$    |
| $X^{(1)}(i_3 i_4 k_5 k_6 k_7 k_8   i_0 i_1 i_2)$    | $X^{(1)}(i_5 i_4 k_3 i_7 i_6 i_8   k_2 k_1 k_0)$    |
| $X^{(2)}(k_3 k_4 k_5 k_6 k_7 k_8   i_0 i_1 i_2)$    | $X^{(2)}(i_5 i_4 i_3 i_7 i_6 i_8   k_2 k_1 k_0)$    |
| $X^{(3b)}(k_3 k_4 k_5 k_6 k_7 i_2   i_0 i_1 k_8)^*$ | $X^{(3b)}(i_5 i_4 i_3 i_7 i_6 k_2   i_8 k_1 k_0)^*$ |
| $X^{(3a)}(k_3 k_4 k_5 k_6 i_2 k_7   i_0 i_1 k_8)$   | $X^{(3a)}(i_5 i_4 i_3 k_2 i_6 i_7   i_8 k_1 k_0)$   |
| $X^{(3*)}(k_3 k_4 k_5 k_6 k_2 k_1   i_0 k_7 k_8)^*$ | $X^{(3*)}(i_5 i_4 i_3 i_2 i_6 i_1   i_8 i_7 k_0)^*$ |
| $X^{(4a)}(k_3 k_4 k_5 k_1 k_2 k_6   i_0 k_7 k_8)$   | $X^{(4a)}(i_5 i_4 i_3 i_2 i_1 i_6   i_8 i_7 k_0)$   |
| $X^{(4*)}(k_3 k_4 k_5 k_1 k_2 k_0   k_6 k_7 k_8)^*$ | $X^{(4*)}(i_5 i_4 i_3 i_2 i_1 i_0   i_8 i_7 i_6)$   |
| bit reversed  | bit reversed  |
| $x(k_5 k_4 k_3 k_7 k_6 k_8   k_2 k_1 k_0)$          | $x(i_3 i_4 i_5 i_6 i_7 i_8   i_0 i_1 i_2)$          |

**Time Complexity.** The computation costs of PFFTs are currently evaluated by the number of complex multiplications and additions/subtractions. The reason that higher radix FFTs are more efficient than radix-2 FFTs is that the symmetries of trigonometric functions are employed. Hence, in order to analyze the computation complexity of FFTs, the number of real multiplications and additions/subtractions needed must be considered. This is dependent on the order of the twiddle factors.

(A) Radix-2. For radix-2 FFTs, the computation is reduced by  $\omega_{2^n}^{lk \cdot 2^{n-1}} = 1$  or  $-1$ ,  $\omega_{2^n}^{lk \cdot 2^{n-2}} = 1, -1, i$  or  $-i$ , and  $\omega_{2^n}^{lk \cdot 2^{n-3}} = \omega_8^l$ . Thus, complex multiplication with  $\omega_2^l$  and  $\omega_4^l$  requires no computation, and complex multiplication with the twiddle factor  $\omega_8^l$  can be implemented by 2 real multiplications and 2 real additions/subtractions. All multiplications with nontrivial twiddle factors can be implemented by 3 real multiplications and 3 real additions/subtractions. Thus, the first two stages of the FFT require no multiplications, the third stage requires 2 real multiplications and 2 real additions/subtractions per complex multiplication, and the remaining stages use 3 real multiplications and 3 real additions per complex multiplication. One complex addition/subtraction is implemented by 2 real additions/subtractions. The number of operations for computing a radix-2 PFFT is

thus given by:

$$M[PFFT_2]_N = \begin{cases} 2^{n-d} + 3(n-3)2^{n-d-1} & n-d \geq 1 \text{ real} \\ 3n-7 & n-d = 0 \text{ multiplications,} \end{cases} \quad (2a)$$

$$A[PFFT_2]_N = M[PFFT_2]_N + n \cdot 2^{n-d+1} \text{ real additions/subtractions.} \quad (2b)$$

Thus, the time complexity of a radix-2 unordered PFFT pair algorithm is

$$T_{2d1}(n, d) = T_{\text{comp-2}} + T_{\text{comm-2d1}} \quad (3)$$

where

$$T_{\text{comp-2}} = 2(M[PFFT_2]_N \cdot \gamma_1 + A[PFFT_2]_N \cdot \gamma_2),$$

$\gamma_1$ : the computation cost of one real \* (in flops)

$\gamma_2$ : the computation cost of one real +/- (in flops)

$$T_{\text{comm-2d1}} = 2d(\alpha + \beta_1 \cdot 2^{n-d-1} \cdot 8)$$

$2d$ : total number of distance-1 communications per FFT pair

$\alpha$ : start up cost (in flops)

$\beta_1$ : per-byte transmission cost (in flops) for distance-1 communication

$8$ : length of complex number (in bytes)

$2^{n-d-1}$ : number of transmission data points.

(B) Mixed-radix (4-2). The computation analysis for the mixed-radix (4-2) PFFT can be obtained by a similar approach. The discussion of computation costs for the mixed-radix (4-2) PFFTs is divided into two parts:  $n$  is even and  $n$  is odd.

Part 1.  $n$  even: The computation procedure for mixed-radix (4-2) PFFTs is exactly the same as the procedure for the radix-4 PFFTs which uses the properties  $\omega_4^{lk \cdot 4^{r-1}} = \pm 1$  or  $\pm i$  and  $\omega_4^{lk \cdot 4^{r-2}} = \omega_8^{lk}$  when  $lk$  is even. Thus, the computation costs for mixed-radix (4-2) PFFTs is given by  $M[PFFT_4]_N$  real multiplications and  $A[PFFT_4]_N$  real additions/subtractions where,

$$M_{\text{even}}[PFFT_{(4-2)}]_N = M[PFFT_4]_N = \begin{cases} 2^{n-d+1} + 9 \left(\frac{n}{2} - 2\right) 2^{n-d-2} & n-d \geq 2 \\ \frac{9}{2}n - 10 & n-d = 1, 0 \end{cases} \quad (4a)$$

$$A_{\text{even}}[PFFT_{(4-2)}]_N = A[PFFT_4]_N \quad (4b)$$

$$= \begin{cases} 2^{n-d+1} \cdot n + M[PFFT_{\text{mixed-radix}}]_N & n-d \geq 2 \\ 6 \cdot n + M[PFFT_{\text{mixed-radix}}]_N & n-d = 1 \\ 3 \cdot n + M[PFFT_{\text{mixed-radix}}]_N & n-d = 0. \end{cases}$$

Part 2.  $n$  odd: The relationship between the numbers of complex multiplications, additions/subtractions and the type of twiddle factors used in each stage is summarized in Table II.

TABLE II.

The relationship between the number of complex multiplications, additions/subtractions and type of twiddle factors for mixed-radix (4-2) PFFTs,  $n$  odd.

| Stage $t$ | The number of complex multiplications | Type of twiddle factors     | The number of complex multiplications/subtractions |
|-----------|---------------------------------------|-----------------------------|--|
| 0         | $2^{n-d}$                             | $\omega_{4^p}^{lk4^{p-1}}$  | $2 \cdot 2^{n-d}$                                  |
| 1         | $2^{n-d}$                             | $\omega_{4^p}^{lk4^{p-2}}$  | $2 \cdot 2^{n-d}$                                  |
| 2         | $2^{n-d}$                             | $\omega_{4^p}^{lk4^{p-3}}$  | $2 \cdot 2^{n-d}$                                  |
| $\vdots$  | $\vdots$                              | $\vdots$                    | $\vdots$   |
| $p-1$     | $2^{n-d}$                             | $\omega_{4^p}^{lk}$         | $2 \cdot 2^{n-d}$                                  |
| $p$       | $2^{n-d-1}$                           | $\omega_{4^p}^{lk \cdot 2}$ | $2^{n-d}$  |

Note:  $n = 2p + 1$ ,  $l = 0, 1, 2, 3$  and  $k = 0, \dots, 4^l - 1$ .

From the previous table, it is known that the number of operations required for computing mixed-radix (4-2) PFFTs, when  $n$  is odd, is:

$$O_{M_{\text{odd}}}[PFFT_{(4-2)}]N = \left(\frac{n-1}{2}\right) \cdot 2^{n-d} + 2^{n-d-1} \text{ complex multiplications and} \quad (5a)$$

$$O_{A_{\text{odd}}}[PFFT_{(4-2)}]N = \left(\frac{n-1}{2}\right) \cdot 2^{n-d+1} + 2^{n-d} \text{ complex additions/subtractions} \quad (5b)$$

Again the number of operations can be reduced by the properties used in (A). Thus, no multiplication is required for stage 0. In stage 1, 8 real multiplications and 8 real additions/subtractions are required in evaluating every group of 4 points. There are 9 real multiplications and 9 real additions/subtractions required to evaluate each group of 4 points. Hence, the numbers of real multiplications and additions/subtractions for computing mixed-radix (4-2) PFFTs when  $n$  is odd can be reduced as follows:

$$M_{\text{odd}}[PFFT_{(4-2)}]N = \begin{cases} (9n-17)2^{n-d-3} & n-d \geq 2 \\ \frac{9}{2}n - \frac{23}{2} & n-d = 1, 0 \end{cases} \quad (6a)$$

and

$$A_{\text{odd}}[PFFT_{(4-2)}]N = \begin{cases} n \cdot 2^{n-d+1} + M_{\text{odd}}[PFFT_{(4-2)}]N & n-d \geq 2 \\ 6n-2 + M_{\text{odd}}[PFFT_{(4-2)}]N & n-d = 1 \\ 3n-1 + M_{\text{odd}}[PFFT_{(4-2)}]N & n-d = 0. \end{cases} \quad (6b)$$

Thus, the numbers of real multiplications and additions/subtractions for computing mixed-radix (4-2) PFFTs are

$$M[PFFT_{(4-2)}]N = \begin{cases} M_{\text{odd}}[PFFT_{(4-2)}]N & \text{if } n = 2k + 1 \\ M_{\text{even}}[PFFT_{(4-2)}]N & \text{if } n = 2k \end{cases} \quad (7a)$$

and

$$A[PFFT_{(4-2)}]N = \begin{cases} A_{\text{odd}}[PFFT_{(4-2)}]N & \text{if } n = 2k + 1 \\ A_{\text{even}}[PFFT_{(4-2)}]N & \text{if } n = 2k, \end{cases} \quad (7b)$$

where  $k$  is any positive integer.

By observing the distance-1 forward and inverse mixed-radix (4-2) PFFT algorithms in Tables I and II, the use of partial-exchange  $i$ -cycles again induces no interprocessor communication and no data rearrangement over the first  $\lfloor \frac{n-d}{2} \rfloor$  stages. After the first  $\lfloor \frac{n-d}{2} \rfloor$  stages,  $d$  distance-1 communications of packets of size  $2^{n-d-1}$  are required.

Thus,  $2d$  distance-1 communications are required for executing a mixed-radix distance-1 PFFT pair. Also, the equations (7a) and (7b) give the computation cost of mixed-radix PO2 PFFTs,  $M[PFFT_{(4-2)}]N$  and  $A[PFFT_{(4-2)}]N$ . Thus, the time complexity,  $T_{md1}(n, d)$ , can be given as follows:

$$T_{md1}(n, d) = T_{\text{comp-md1}} + T_{\text{comm-md1}} \quad (8)$$

where

$$T_{\text{comp-md1}} = 2(M[PFFT_{\text{mixed-radix}(4-2)}]N \cdot \gamma_1 + A[PFFT_{\text{mixed-radix}(4-2)}]N \cdot \gamma_2) \text{ and} \\ T_{\text{comm-md1}} = 2 \cdot d \cdot (\alpha + \beta 1 \cdot 2^{n-d-1} \cdot \delta).$$

**Comparison of Mixed-radix (4-2) with radix-2.** Comparing  $T_{2d1}(n, d)$  with  $T_{md1}(n, d)$ , it is apparent that both PFFT pairs require the same number of distance-1 communications of the same packet size. This implies that the choice of the more efficient PFFT pair between radix-2 and mixed-radix PFFT pair depends on their respective computation costs:

**Theorem 1.** The mixed-radix (4-2) PFFTs require less computation than the radix-2 PFFTs except in the following cases:

- $n-d = 1, 0, n \geq 2$  and  $n$  is even for real multiplications
- $n-d = 1, 0, n \geq 3$  and  $n$  is odd for real multiplications
- $n-d = 1$  and  $n$  is even for real additions/subtractions
- $n-d = 1, n$  is odd,  $n > 1$  for real additions/subtractions
- $n-d = 0, n$  is even,  $n > 1$  for real additions/subtractions
- $n-d = 0, n$  is odd,  $n > 2$  for real additions/subtractions
- $(n = 2, d = 0)$  or  $(n = 3, d = 0)$  or  $(n = 3, d = 1)$  for real multiplications and additions/subtractions.

*Proof.* The proof can be accomplished by examining the differences between the number of operations (real multiplications and real additions) for the two different PFFTs. Details can be found in [7].

From Theorem 1, it is observed that the number of operations for the mixed-radix (4-2) PFFTs is much less for large  $n$  or large  $n-d$  provided  $n-d \geq 2$ . From Theorem 1, it is known that the mixed-radix (4-2) PFFTs require less computation when  $n-d \geq 2$  and  $n > 3$ . Hence, it is concluded that the distance-1 mixed-radix (4-2) PFFT pair is faster than the distance-1 radix-2 PFFT pair for  $n-d \geq 2$  and  $n > 3$ .

It is further concluded that the distance-1 mixed-radix (4-2) unordered PFFT pair is the choice among all the distance-1 unordered PFFT pairs when  $n - d \geq 2$  and  $n > 3$ . Actually the condition,  $n - d \geq 2$  and  $n > 3$ , is very reasonable for each processor should own at least 4 data elements to gain the complete advantage of using the mixed-radix (4-2) PFFT algorithms and the number of data points should be large to gain the advantage of parallelism.

**Unordered Distance-2 Parallel FFT Pairs.** For all the distance-2 PO2 PFFT algorithms, any two digits in the address id position can be chosen as a pivot element set. As for the distance-1 algorithms, the choice of pivot element set affects the development of PFFT algorithms. Again, the left-most two digits or right-most two digits are preferred since they allow for portable algorithms. To be consistent choose the right-most two digits as the pivot element set. Note that for all the unordered forward distance-2 PO2 PFFT algorithms, the index sequence of the pivot element set should be exchanged in ascending order from left to right in process id position and descending order from left to right in address id position. For all the unordered inverse distance-2 PO2 PFFT algorithms, the index sequence of the pivot element set should be exchanged in descending order from left to right in both process id and address position.

The communication patterns for distance-2 unordered PFFT algorithms are (a) distance-1 communication with packets of size  $2^{n-d-1}$  data elements, and (b) distance-2 communication with packets of size  $2^{n-d-2}$  data elements. Figure 1 shows the examples for the communication patterns (a) and (b) of only one processor on a hypercube with 8 processors.

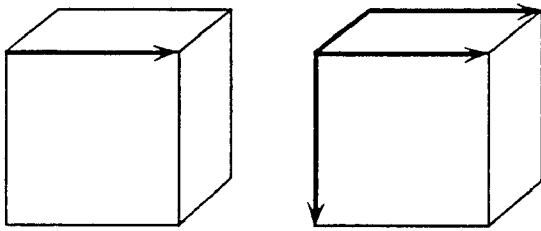


Figure 1.

**Algorithms.** Algorithms for the unordered distance-2 forward and inverse PFFT pairs with radix-2 and mixed-radix (4-2) are discussed here. Details are given in [7]. In each algorithm the manner in which it proceeds depends on the relation between  $n$  and  $d$ . In particular, for radix-2 unordered forward and inverse PFFT algorithms, all the communications are of distance-2 with packets of size  $2^{n-d-2}$  data elements except when  $d$  is odd, then one more distance-1 communication is required. For the mixed-radix (4-2) unordered forward and inverse PFFT algorithms all communications are distance-2 except that one distance-1 communication is required when  $n - d$  is odd and an additional distance-1 communication is required when  $n$  is odd. Examples of the unordered distance-2 PFFT pairs are given in Table III for radix-2 and Table IV for mixed-radix (4-2). The notation is the same as that adopted in the previous section. In addition, 'e' means reordering inside each node is performed; 'f' indicates the stage is calculated by a 2-element partial i-cycle.

TABLE III.

Unordered distance-2 radix-2 PFFT algorithms ( $n = 9, d = 6$ ).

| (a) forward  | (b) inverse  |
|--|--|
| $x(i_6 i_7 i_8   i_0 i_1 i_2 i_3 i_4 i_5)$             | $x(k_6 k_7 k_8   k_5 k_4 k_3 k_2 k_1 k_0)$             |
| $X^{(0)}(i_6 i_7 k_8   i_0 i_1 i_2 i_3 i_4 i_5)$       | $X^{(0)}(k_6 k_7 i_8   k_5 k_4 k_3 k_2 k_1 k_0)$       |
| $X^{(1)}(i_6 k_7 k_8   i_0 i_1 i_2 i_3 i_4 i_5)$       | $X^{(1)}(k_6 i_7 i_8   k_5 k_4 k_3 k_2 k_1 k_0)$       |
| $X^{(2)}(k_6 k_7 k_8   i_0 i_1 i_2 i_3 i_4 i_5)$       | $X^{(2)}(i_6 i_7 i_8   k_5 k_4 k_3 k_2 k_1 k_0)$       |
| $X^{(3f)}(k_6 k_5 i_4   i_0 i_1 i_2 i_3 k_7 k_8)^{**}$ | $X^{(3f)}(i_6 i_5 k_4   i_8 i_7 k_3 k_2 k_1 k_0)^{**}$ |
| $X^{(4)}(k_6 k_5 k_4   i_0 i_1 i_2 i_3 k_7 k_8)$       | $X^{(4)}(i_6 i_5 i_4   i_8 i_7 k_3 k_2 k_1 k_0)$       |
| $X^{(5a)}(k_4 k_5 k_6   i_0 i_1 i_2 i_3 k_7 k_8)$      | $X^{(5a)}(i_4 i_5 i_6   i_8 i_7 k_3 k_2 k_1 k_0)$      |
| $X^{(5f)}(k_4 k_3 i_2   i_0 i_1 k_5 k_6 k_7 k_8)^{**}$ | $X^{(5f)}(i_4 i_3 k_2   i_8 i_7 i_6 i_5 k_1 k_0)^{**}$ |
| $X^{(6)}(k_4 k_3 k_2   i_0 i_1 k_5 k_6 k_7 k_8)$       | $X^{(6)}(i_4 i_3 i_2   i_8 i_7 i_6 i_5 k_1 k_0)$       |
| $X^{(7a)}(k_2 k_3 k_4   i_0 i_1 k_5 k_6 k_7 k_8)$      | $X^{(7a)}(i_2 i_3 i_4   i_8 i_7 i_6 i_5 k_1 k_0)$      |
| $X^{(7f)}(k_2 k_1 i_0   k_3 k_4 k_5 k_6 k_7 k_8)^{**}$ | $X^{(7f)}(i_2 i_1 k_0   i_8 i_7 i_6 i_5 i_4 i_3)^{**}$ |
| $X^{(8)}(k_2 k_1 k_0   k_3 k_4 k_5 k_6 k_7 k_8)$       | $X^{(8)}(i_2 i_1 i_0   i_8 i_7 i_6 i_5 i_4 i_3)$       |
| bit reversed   | bit reversed   |
| $x(k_6 k_7 k_8   k_5 k_4 k_3 k_2 k_1 k_0)$             | $x(i_6 i_7 i_8   i_0 i_1 i_2 i_3 i_4 i_5)$             |

TABLE IV.

Unordered distance-2 mixed-radix PFFT algorithms ( $n = 9, d = 5$ ).

| (a) forward  | (b) inverse  |
|--|--|
| $x(i_5 i_6 i_7 i_8   i_0 i_1 i_2 i_3 i_4)$             | $x(k_7 k_5 k_6 k_8   k_4 k_3 k_2 k_1 k_0)$             |
| $X^{(0)}(i_5 i_6 k_7 k_8   i_0 i_1 i_2 i_3 i_4)$       | $X^{(0)}(i_7 k_5 k_6 i_8   k_4 k_3 k_2 k_1 k_0)$       |
| $X^{(1)}(k_5 k_6 k_7 k_8   i_0 i_1 i_2 i_3 i_4)$       | $X^{(1)}(i_7 i_5 i_6 i_8   k_4 k_3 k_2 k_1 k_0)$       |
| $X^{(2f)}(k_5 k_6 k_4 k_3   i_0 i_1 i_2 k_7 k_8)^{**}$ | $X^{(2a)}(i_6 i_5 i_8 i_7   k_4 k_3 k_2 k_1 k_0)$      |
| $X^{(3a)}(k_4 k_3 k_5 k_6   i_0 i_1 i_2 k_7 k_8)$      | $X^{(2f)}(i_6 i_5 i_4 i_3   i_8 i_7 k_2 k_1 k_0)^{**}$ |
| $X^{(3f)}(k_4 k_3 k_2 k_1   i_0 k_5 k_6 k_7 k_8)^{**}$ | $X^{(3a)}(i_4 i_3 i_6 i_5   i_8 i_7 k_2 k_1 k_0)$      |
| $X^{(4a)}(k_1 k_3 k_2 k_4   i_0 k_5 k_6 k_7 k_8)$      | $X^{(3f)}(i_4 i_3 i_2 i_1   i_8 i_7 i_6 i_5 k_0)^{**}$ |
| $X^{(4*)}(k_1 k_3 k_2 k_0   k_4 k_5 k_6 k_7 k_8)^*$    | $X^{(4a)}(i_1 i_3 i_2 i_4   i_8 i_7 i_6 i_5 k_0)$      |
| bit reversed   | $X^{(4*)}(i_1 i_3 i_2 i_0   i_8 i_7 i_6 i_5 i_4)^*$    |
| $x(k_7 k_5 k_6 k_8   k_4 k_3 k_2 k_1 k_0)$             | $X^{(4e)}(i_3 i_2 i_1 i_0   i_8 i_7 i_6 i_5 i_4)$      |
|  | bit reversed   |
|  | $x(i_5 i_6 i_7 i_8   i_0 i_1 i_2 i_3 i_4)$             |

**Time complexity of distance-2 unordered PO2 PFFT algorithms.** A) Radix-2. The radix-2 unordered distance-2 forward and inverse PFFT algorithms apply the partial exchange  $i$ -cycle. During the first  $n - d$  stages there is no interprocessor communication and no data rearrangement within processors. After the first  $(n - d)$  stages, the algorithm uses a pattern in which communication is followed by computation of two stages of the FFT. If  $d$  is odd, then one more distance-1 communication is required at the end (when  $ix = 0$ ).

By observation,  $(d - 1)/2$  distance-2 communications of length  $2^{n-d-2}$  and one distance-1 communication of length  $2^{n-d-1}$  are required when  $d$  is odd and  $d/2$  distance-2 communications of length  $2^{n-d-2}$  are required when  $d$  is even. Hence, in total,  $\lceil \frac{d}{2} \rceil$  communications are required for both the forward and inverse unordered radix-2 PFFT algorithms. As discussed, the computation costs for radix-2 PFFTs are independent of the communication pattern. Thus, the time complexity of the distance-2 unordered radix-2 PFFT pair is:

$$T_{2d2}(n, d) = T_{\text{comp-}2d2} + T_{\text{comm-}2d2} \quad (9)$$

where

$$\begin{aligned}
T_{\text{comp-2d2}} &= 2(M[\text{PFFT}_{\text{radix-2}}]N \cdot \gamma_1 \\
&\quad + A[\text{PFFT}_{\text{radix-2}}]N \cdot \gamma_2) \quad \forall d, \\
T_{\text{comm-2d2}} &= \begin{cases} d \cdot TS_2 & \text{if } d = 2l \\ (d-1) \cdot TS_2 + 4TS_1 & \text{if } d = 2l+1, \end{cases} \\
TS_1 &= \alpha + \beta_1 \cdot 2^{n-d-1} \cdot 8, \\
TS_2 &= \max\{\alpha + \beta_2 \cdot 2^{n-d-2} \cdot 8, \\
&\quad \alpha + \delta_2 + \delta_1 + \beta_1 \cdot 2^{n-d-2} \cdot 8\},
\end{aligned}$$

$l$  is any positive integer,  $\delta_1$  is the time delay due to sending one distance-1 communication, and  $\delta_2$  is the time delay due to sending one distance-2 communication.

B) Mixed-radix (4-2). Over the first  $\lfloor \frac{n-d}{2} \rfloor$  stages of the mixed-radix unordered forward and inverse distance-2 PFFT algorithms, the use of partial-exchange 2-element  $i$ -cycles again means that no communication and no data rearrangement within processors occur. Right after the first  $\lfloor \frac{n-d}{2} \rfloor$  stages, one distance-1 communication followed by computation of one stage of the FFT is required if  $(n-d)$  is odd. The algorithm then proceeds with a pattern in which one distance-2 communication and one PFFT stage computation are carried out in turn until the end ( $idx = 0$ ). If  $n$  is odd, then one more distance-1 communication while computing one PFFT radix-2 stage is executed at the end.

The total number of communications is therefore determined by the structure of the algorithm, again dependent on  $n$  and  $d$ :

- i) if  $d = 2l + 1$ ,  $(d-1)/2$  distance-2 communications of length  $2^{n-d-2}$  and one distance-1 communication of length  $2^{n-d-1}$  are required,
- ii) if  $d = 2l$  and  $n = 2k$ ,  $d/2$  distance-2 communications of length  $2^{n-d-2}$  are required, and
- iii) if  $d = 2l$  and  $n = 2k + 1$ ,  $(d-2)/2$  distance-2 communications of length  $2^{n-d-2}$  and two distance-1 communications of length  $2^{n-d-1}$ , where  $l$  and  $k$  are any integers, are required to implement a forward or inverse unordered mixed-radix (4-2) PFFT.

In total,  $\lfloor \frac{d}{2} \rfloor$  communications are required for  $n$  even and  $\lfloor \frac{d+1}{2} \rfloor$  communications are required for  $n$  odd. Using the computation cost of mixed-radix (4-2) PO2 PFFTs, we have the time complexity  $T_{md2}(u, d)$  as follows:

$$T_{md2}(n, d) = T_{\text{comp-md2}} + T_{\text{comm-md2}} \quad (10)$$

where

$$\begin{aligned}
T_{\text{comp-md2}} &= 2(M[\text{PFFT}_{(4-2)}] \cdot \gamma_1 + A[\text{PFFT}_{(4-2)}] \cdot \gamma_2), \\
T_{\text{comm-md2}} &= \begin{cases} (d-1) \cdot TS_2 + 2 \cdot TS_1 & d = 2l + 1 \\ d \cdot TS_2 & d = 2l \text{ and } n = 2k \\ (d-2) \cdot TS_2 + 4 \cdot TS_1 & d = 2l \text{ and } n = 2k + 1, \end{cases}
\end{aligned}$$

and  $l, k$  are any positive integers.

**Comparison amongst distance-2 unordered PO2 PFFT algorithms.** In order to compare the time complexities of the unordered distance-2 radix-2 PFFT pair and 2 mixed-radix PFFT pairs, we subtract  $T_{md2}$  from  $T_{2d2}$ :

$$\begin{aligned}
T_{2d2}(n, d) - T_{md2}(n, d) &= 2[S_3 \cdot \gamma_1 + S_4 \cdot \gamma_2] \\
&\quad + \begin{cases} 0 & \text{if } d = 2l + 1 \text{ or } d = 2l \text{ and } k = 2k, \\ 2 \cdot TS_2 - 4 \cdot TS_1 & \text{if } d = 2l \text{ and } k = 2k + 1, \end{cases} \quad (11)
\end{aligned}$$

where  $l, k$  are any positive integers,

$$S_3 = M[\text{PFFT}_{\text{radix-2}}]N - M[\text{PFFT}_{\text{mixed-radix}(4-2)}]N, \text{ and}$$

$$S_4 = A[\text{PFFT}_{\text{radix-2}}]N - A[\text{PFFT}_{\text{mixed-radix}(4-2)}]N.$$

By Theorem 1,  $T_{2d2}(n, d) - T_{md2}(n, d) > 0$  when i)  $d$  is odd and ii) both  $d$  and  $n$  are even provided  $n-d \geq 2$  and  $n > 3$  since  $S_3$  and  $S_4$  are positive as  $n-d \geq 2$  and  $n > 3$ . Hence the mixed-radix (4-2) is faster than the radix-2 unordered distance-2 PFFT pair for most reasonable cases.

When  $n$  is odd and  $d$  is even, the value of  $T_{2d2}(n, d) - T_{md2}(n, d)$  depends on both the computation cost,  $S_3\gamma_1 + S_4\gamma_2$ , and the communication cost,  $TS_2 - 2 \cdot TS_1$ . This heavily depends on the design of the machine for the Intel iPSC system. The communication system is closed-channel. In this case, the value of  $TS_2 - 2 \cdot TS_1$  is nonnegative. Thus, the value of  $T_{2d2}(n, d) - T_{md2}(n, d)$  is positive for  $n-d \geq 2$  and  $n > 3$ . For an open channelled machine, such as the CM-5, the sign of  $T_{2d2}(n, d) - T_{md2}(n, d)$  depends on the relationship between the computation speeds  $TS_1$  and  $TS_2$  and the communication speed. Hence no immediate conclusion can be drawn.

**Comparison of distance-1 versus distance-2 unordered parallel FFT pairs.** From the previous sections, it is known that an unordered mixed-radix PFFT pair is faster than an unordered radix-2 PFFT pair in most of the cases for both distance-1 and distance-2 algorithms. But are distance-2 algorithms faster than their comparable distance-1 algorithms? This depends on the communication costs.

Comparing by radix:

(A) Radix-2. Comparing the time costs for the unordered radix-2 distance-1 and distance-2 PFFT pairs, gives

$$\begin{aligned}
T_{2d1}(n, d) - T_{2d2}(n, d) &= \begin{cases} 2[(d-1) \cdot TS_1 - (\frac{d-1}{2}) \cdot TS_2] & \text{if } d = 2l + 1, \\ 2[d \cdot TS_1 - \frac{d}{2} \cdot TS_2] & \text{if } d = 2l, \end{cases} \quad (12)
\end{aligned}$$

where  $l$  is any positive integer. Obviously, the sign of  $T_{2d1}(n, d) - T_{2d2}(n, d)$  is positive if and only if  $2TS_1 > TS_2$ .

Hence, the radix-2 unordered distance-2 PFFT pair is more efficient than the radix-2 unordered distance-1 PFFT pair if and only if  $2 \cdot TS_1 > TS_2$ . This depends on the hardware chip design and the transmission strategy used by parallel machines.

(B) Mixed-radix (4-2). Comparing the time costs for an unordered mixed-radix (4-2) distance-1 and distance-2 PFFT pair, gives

$$T_{md1(n,d)} - T_{md2(n,d)} = \begin{cases} 2[(d-1)TS1 - \frac{(d-1)}{2} \cdot TS2] & \text{if } d = 2l + 1 \\ 2[(d-2)TS1 - \frac{(d-2)}{2} TS2] & \text{if } d = 2l \text{ and } n = 2k + 1 \\ 2[dTS1 - \frac{d}{2} TS2] & \text{if } d = 2l \text{ and } n = 2k, \end{cases} \quad (13)$$

where  $l$  and  $k$  are any positive integers. Again,  $T_{md1(n,d)} - T_{md2(n,d)} > 0$  if and only if  $2TS1 > TS2$ . And conclusions as in (A).

From the above results, it is concluded that if the time cost for a distance-1 communication of length  $2^{n-d-1}$ ,  $TS1$ , is greater than half the time cost for a distance-2 communication of length  $2^{n-d-2}$ ,  $\frac{1}{2}TS2$ , then the unordered distance-2 PFFT pairs should be considered. Otherwise, the unordered distance-1 PFFT pairs should be considered.

**Conclusions.** From the comparisons, it is found that the distance-1 mixed-radix (4-2) unordered PFFT pair is faster than the distance-1 radix-2 unordered PFFT pair when  $n - d \geq 2$  and  $n > 3$ . It is further concluded that when  $2 \cdot TS1 > TS2$ , the distance-2 mixed-radix (4-2) PFFT pair is the choice among all unordered PFFT pairs provided  $n - d \geq 2$  and  $n > 3$  when choosing either  $d$  odd or both  $d$  and  $n$  even. Note that the time for  $TS1$  and  $TS2$  depends on the speed of the communication channel and the routing strategy provided by the machine. In the real-world,  $2TS1 > TS2$  is one of the design goals of the hardware communication which makes distance-2 algorithms useful in applications. We also apply the same ideas of using various radices and distances to develop the ordered PO2 PFFT algorithms [7, 8].

The first author's current address is: Department of Geological Sciences, University of British Columbia, 6339 Stores Road, Vancouver, B. C. V6T 1Z4.

## REFERENCES

- [1] Nussbaumer, H. J. Fast fourier transform and convolution algorithms, 2nd edition, Springer-Verlag, 1982.
- [2] Renaut, R. A., and Woo, M. Parallel pseudospectral methods for the solution of the wave equation. In Fitzgibbon, W. E., and Wheeler, M. F. (Eds.). *Wave Propagation and Inversion*. SIAM, Philadelphia, 1992, pp. 124-134.
- [3] Swarztrauber, P. N. Multiprocessor FFT's. *Parallel Computing* 5 (1987), 197-210.
- [4] Woo, M. Parallel pseudospectral methods on hypercubes. Master's thesis, Arizona State University, 1989.
- [5] Woo, M., and Renaut, R. A. Parallel power-of-two FFTs on hypercubes. *Supercomputing '91*, 754-763.
- [6] Woo, M. and Renaut, R. A. A performance evaluation of parallel FFTs on a hypercube. Arizona State University, Technical Report #130, January 1991.
- [7] Woo, M. Parallel power-of-two fast Fourier transform on a hypercube. Ph.D. Dissertation, Arizona State University, 1992.
- [8] Woo, M., and Renaut, R. A. Ordered parallel distance-1 and distance-2 FFT algorithms of radix 2 and mixed-radix (4-2), submitted to SIAM Journal on Computing, 1993.