Best and worst mappings for the omega network

by Ž. Cvetanović

This paper presents a study of the best and worst mappings for the omega network proposed by D. H. Lawrie in 1975. We identify mappings that produce no conflicts in the network and mappings that produce a maximum number of conflicts. The analysis of mappings for some typical applications shows that an initial allocation of data to memory modules determines the contention within the network for all iterations of the algorithm. For the case of the FFT and the bitonic sort algorithm executed on a shared-memory architecture, we prove that if no conflicts are produced during the first iteration of the algorithm, then no conflicts are produced during any other iteration. Moreover, if a maximum number of conflicts are produced during the first iteration, then a maximum number of conflicts are produced during all other iterations of the algorithm. For the *d*-dimensional grid computations where communication is required with 2*d* nearest neighbors, we prove that if the initial allocation produces no conflicts within the network, then communication with all the neighbors is conflict-free. Also, if the initial allocation produces a maximum number of conflicts, then communication with all the neighbors is maximum-conflict. We show that the omega network cannot produce without conflicts some of the bit-permute mappings such as the perfect shuffle and the bit reversal. The network can produce both of these mappings provided that data items are accessed from memories according to a specific skewed scheme.

1. Introduction

The analysis by Cvetanović [1] of the performance of iterative algorithms executed on both dataflow and shared-memory parallel architectures shows that the method for allocating data to local memories in a dataflow architecture or to global memories in a shared-memory architecture is significant for controlling the contention within the interconnection network. In particular, we have shown that for the omega network proposed by Lawrie [2], the allocation affects the communication time by a factor of $O(\sqrt{N})$, where N is the number of processors. Conflicts within the interconnection network become a significant source of overhead in dataflow architectures that exploit pipelining of operations, since the delay at a single stage of the pipeline is spread to the whole system. A similar effect is observed in shared-memory architectures executing synchronous parallel algorithms, where the execution time between synchronization points is determined by the processor that takes longest to finish.

In this paper, we identify the allocations that do not produce any conflicts within the network and the allocations that produce a maximum number of conflicts. We derive expressions for the number of allocations of each type and characterize their properties. For an N-processor system, the number of conflict-free allocations is shown to be $\sqrt{N}^N$, while the number of maximum-conflict allocations is $(\sqrt{N}!)^{\sqrt{N}+1}$.

For the case of the FFT and bitonic sort algorithm executed on a shared-memory architecture, we prove that if data items are allocated to memory modules such that no conflicts are produced during the first iteration of the

algorithm, then no conflicts are produced during any other iteration. Moreover, if data items are allocated to memory modules such that a maximum number of conflicts are produced during the first iteration of the algorithm, then a maximum number of conflicts are produced during all other iterations.

We extend the analysis to d-dimensional grid computations where at each grid point, communication is required with $2d$ nearest neighbors. Examples of applications with $2d$-neighbor communication include grid-type computations such as the Poisson equation on a d-dimensional grid, the weather-prediction model, and turbulence modeling in fluid flow. For these types of communication patterns, we prove that if the initial allocation produces no conflicts within the network, then communication with all the neighbors is also conflict-free. Moreover, if the initial allocation produces a maximum number of conflicts, then a maximum number of conflicts are produced with all the neighbors.

Finally, we prove that some important bit-permute mappings such as the perfect shuffle and the bit reversal cannot be achieved without conflicts through the network. However, if data items are accessed from memories in a skewed manner, then the omega network can produce both the initial mapping and the perfect shuffle and bit reversal of this mapping.

The next section presents the background and states the assumptions. In Section 3, we define formally the property that holds for the best and worst mappings. In Section 4, we describe processor-memory mappings for the FFT, and we extend the analysis to the bitonic sort algorithm in Section 5. The analysis of mappings for the nearest-neighbor communication appears in Section 6. In Section 7, we analyze some of the bit-permute mappings. The conclusions appear in the last section.

2. Background and assumptions

The interconnection network we study in this paper is Lawrie's omega network [2], with N inputs and N outputs and $(N/2)\log N$ two-input–two-output switches interconnected by a perfect-shuffle connection where N is a power of 2. We assume that each switch has associated infinite-length queues for maintaining multiple messages. Each switch can be in one of four states, two of which are conflicting states and the other two nonconflicting. An example of a 16×16 omega network is shown in **Figure 1.**

Stone [3] has shown that some important algorithms such as the FFT, polynomial evaluation, and sorting can be efficiently implemented using the shuffle-exchange interconnection among processors. Lawrie [2] has proved that the omega network can produce without conflicts many of the important accesses and alignments of rows, columns, and diagonals in an array processor. In particular, he has shown that if the omega network can produce a conflict-free one-to-one mapping P_N from N inputs S_i to N outputs D_i represented by

$$P_N = \{(S_i, D_i) \mid 0 \le i < N\},$$

then it can produce a uniform shift of a conflict-free mapping $P_N + a$ represented by

$$P_N + a = \{(S_i + a(\mathrm{mod}N), D_i) \mid (S_i, D_i) \in P_N\}.$$

We use Lawrie's terminology, where $S = s_1 s_2 \cdots s_n$ represents the binary address of the source, $D = d_1 d_2 \cdots d_n$ is the address of the destination, and $n = \log N$. Lawrie [2] has shown that the switch position in network stage k for the path between the source S and the destination D is determined by the $(\log N - k)$ least significant bits (LS bits) of the source S and the k most significant bits (MS bits) of the destination D. **Figure 2** illustrates a method for determining the path between the source $S = 0100$ and the destination $D = 1101$. In the first network stage, the switch position is determined by three LS bits from the source address and one MS bit from the destination address. In order to determine the switch address at each successive stage, the number of bits from the source is decremented by one and the number of bits from the destination is incremented by one.

In the sections that follow, we assume that the number of processors N is equal to the problem size M. If, however, $N < M$, which is usually the case, our results still hold, since for all algorithms analyzed here, the same communication pattern is maintained by increasing the problem size.

As a specific case of an architecture, we consider a shared-memory parallel computer where the omega network connects N processors to N memory modules and where each data access requires traversing the network. In Cvetanović [1], we show how the results obtained here can be applied to other multiple-processor architectures.

The network is traversed twice for all load requests, since at the beginning of each data access the request is sent from a processor to a memory module and then data items are transmitted in the opposite direction. For the omega network from Figure 1, the number of conflicts in network switches is the same in both directions. This is true since there is a unique path from any source to any destination, and thus the same set of network switches is traversed in both directions. Therefore, we analyze only mappings from processors to memories.

3. Properties of the best and worst mappings

In this section, we identify a property common to all mappings that can be produced without conflicts (conflict-free mappings), and a property common to all mappings that yield a maximum number of conflicts (maximum-conflict mappings). We also determine the number of mappings of each type as a function of N.

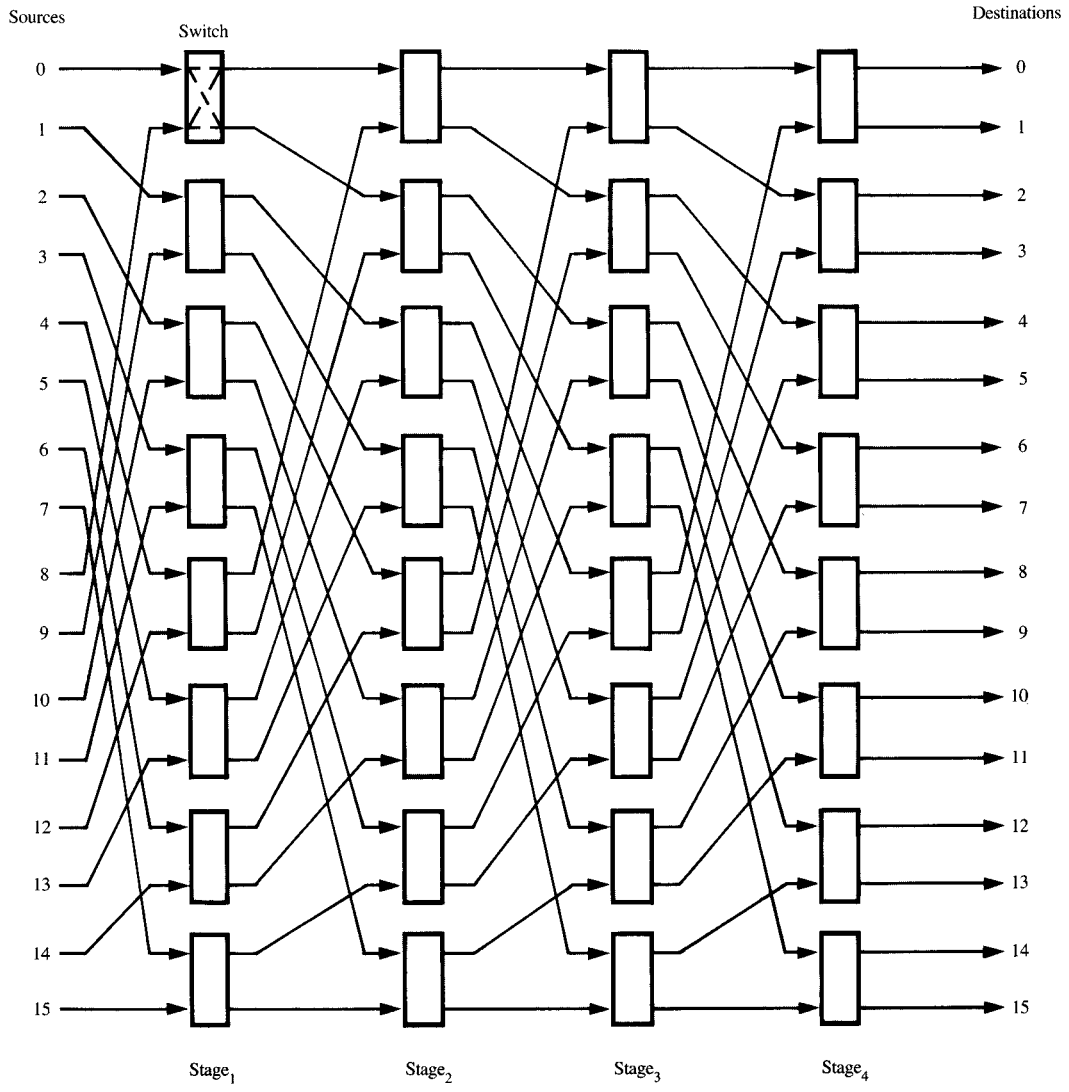

Figure 1

A 16×16 omega network.

Definition 1 The conflict-free mapping is defined as a mapping for which two or more messages are not in conflict at the same switch. For conflict-free mappings, all N paths from N sources to N destinations are disjoint.

Definition 2 The maximum-conflict mapping is defined as a mapping for which N messages are in conflict at $2^{\lceil (\log N)/2 \rceil}$ switches of the $\lfloor (\log N)/2 \rfloor$ network stage, such that $2^{\lfloor (\log N)/2 \rfloor}$ messages are in conflict at the same switch.

Lang [4] has proved that the maximum number of conflicts for the shuffle-exchange network is equal to $2^{\lfloor (\log N)/2 \rfloor} = O(\sqrt{N})$. Consequently, for the omega network which applies the shuffle interconnection between stages, the maximum number of conflicts is equal to $2^{\lfloor (\log N)/2 \rfloor}$, and it occurs in the middle stage of the network (bottleneck stage).

Definition 3 The distance between destinations (sources) i and j, where $i > j$, is defined as the integer *distance* which is a power of two and for which $(i - j) \bmod distance = 0$.

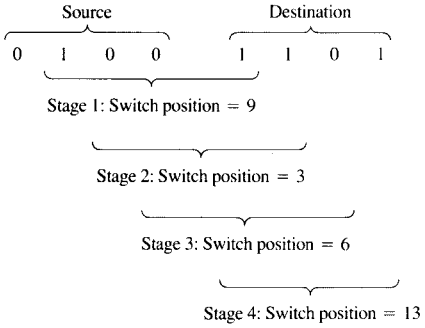

Figure 2

Determining a switch position in each network stage.

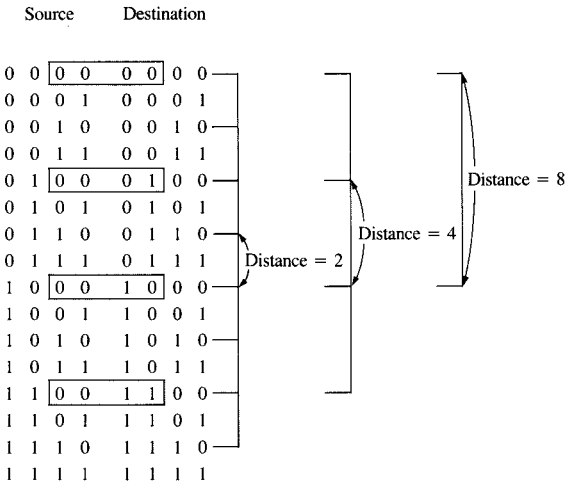

Figure 3

Identity mapping for $N = 16$ processors.

• *Property of conflict-free mappings*

The next claim identifies the property of conflict-free mappings.

Claim 1 A mapping belongs to the class of conflict-free mappings iff the k MS bits cannot be identical for all sets of $N/2^{\log N-k}$ destination addresses at a distance equal to $2^{\log N-k}$, where $k = 1, 2, \cdots, \log N$ is the network stage number.

Proof As demonstrated earlier, a switch position in the stage k is determined by the $(\log N - k)$ LS bits of the source address and the k MS bits of the destination address. Since the consecutive source addresses are in increasing order, the $(\log N - k)$ LS bits from all source addresses at a distance equal to $2^{\log N-k}$ for stage k are identical. Therefore, in order to avoid having two or more switch positions be identical for stage k, the k MS bits from all destinations at this distance must not be the same. Note also that for mappings which produce conflicts within the network, this property does not hold, since the property implies that all N switch positions at each network stage are different. □

As an example, let us demonstrate the property of conflict-free mappings on the identity mapping for $N = 16$ from **Figure 3.** Lawrie [2] has proved that the identity mapping belongs to the class of conflict-free mappings. The first column in Figure 3 represents binary addresses of the sources, while the second column shows binary addresses of the destinations. The rows represent the source address and the address of the destination referenced by this source. At the first stage of the network ($k = 1$), only one MS bit is different for all destination addresses with distance equal to $2^{4-1} = 8$. Note that the addresses of destinations D_0 and D_8 differ in their first MS bit. The same holds for the sets of destinations including D_1 and D_9, D_2 and D_{10}, etc. At the second stage, all destination addresses with distance equal to $2^{4-2} = 4$ have two MS bits different. For example, destinations D_0, D_4, D_8, D_{12} differ in two MS bits. The same holds for the set of destinations including D_1, D_5, D_9, D_{13}, etc. For example, the switch addresses for destinations D_0, D_4, D_8, D_{12} at the second network stage are marked by four windows in Figure 3 and all subsequent figures. Note that two LS bits of source addresses S_0, S_4, S_8, S_{12} are the same, while two MS bits of corresponding destination addresses are different. Similarly, we can obtain the number of MS bits that need to be different for destinations in each network stage in order to produce a conflict-free mapping.

Table 1 Number of mappings as a function of N.

N	*Total number of mappings*	*Conflict-free mappings*		*Maximum-conflict mappings*	
		Number	*Percentage*	*Number*	*Percentage*
4	24	16	66.7	8	33.3
16	$2.1 \cdot 10^{13}$	$4.3 \cdot 10^{9}$	0.02	$7.96 \cdot 10^{6}$	$3.8 \cdot 10^{-5}$
64	$1.27 \cdot 10^{89}$	$6.27 \cdot 10^{57}$	$5 \cdot 10^{-30}$	$2.8 \cdot 10^{41}$	$2.2 \cdot 10^{-46}$

The total number of different mappings is equal to $N!$, which is the number of permutations of N different destinations. The number of conflict-free mappings is derived by Chen et al. [5] as the number of settings of $(N/2) \log N$ network switches into one of two states; it is

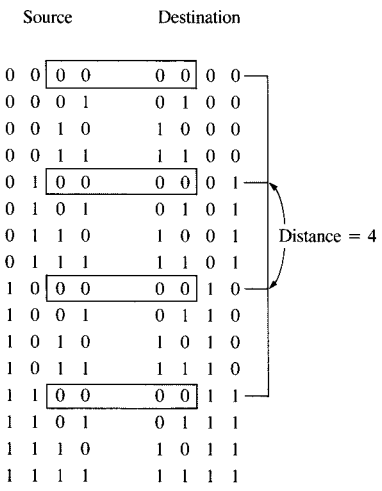

Figure 4

Maximum-conflict mapping for $N = 16$ processors.

equal to $2^{(N/2)\log N} = \sqrt{N}^N$. Note that each switch can be set to one of four states, two of which are the conflict-free states. **Table 1** presents the fraction of all mappings that are conflict-free as a function of N. Note that as N increases, the fraction of conflict-free mappings decreases rapidly.

• *Property of maximum-conflict mappings*

The next claim identifies the property of maximum-conflict mappings, which produce $O(\sqrt{N})$ conflicts in the middle stage of the network.

Claim 2 A mapping belongs to the class of maximum-conflict mappings iff the $\lfloor (\log N)/2 \rfloor$ MS bits are identical for all sets of $N/2^{\lceil (\log N)/2 \rceil}$ destination addresses with distance equal to $2^{\lceil (\log N)/2 \rceil}$.

Proof As stated by Definition 2, a maximum number of conflicts are produced in the $\lfloor (\log N)/2 \rfloor$ network stage. The switch position in this stage is determined by the $\lceil (\log N)/2 \rceil$ LS bits of the source address and the $\lfloor (\log N)/2 \rfloor$ MS bits of the destination address. The $\lceil (\log N)/2 \rceil$ LS bits from source addresses at distance equal to $2^{\lceil (\log N)/2 \rceil}$ for stage $\lfloor (\log N)/2 \rfloor$ are identical. Therefore, in order to have all switch positions with this distance identical, the $\lfloor (\log N)/2 \rfloor$ remaining bits from the destination addresses at this distance must be the same. □

As an example, **Figure 4** shows one of the maximum-conflict mappings for $N = 16$. In this case, the $\lfloor (\log N)/2 \rfloor = 2$ MS bits are identical for all four destinations with distance equal to $\sqrt{N} = 4$. Note that for destinations D_0, D_4, D_8, D_{12}, marked by four windows on this figure, two MS address bits are identical. These four destinations and four corresponding sources determine the address of switch 0. The same holds for the set of destinations including D_1, D_5, D_9, D_{13}, and for two other sets. The maximum number of conflicts in this case is equal to 4 and is found in the bottleneck switches 0, 5, 10 and 15 of the second network stage.

In order to simplify the analysis, we now assume that $N = 2^{2k}$, where k is an integer. The results for other N can be obtained by substituting a floor function for $\sqrt{N}$.

Claim 3 The total number of maximum-conflict mappings is equal to $(\sqrt{N}!)^{\sqrt{N}+1}$.

Proof The number of maximum-conflict mappings is obtained as a product of the number of arrangements of $\sqrt{N}$ different groups each including $\sqrt{N}$ neighboring destinations. The number of arrangements for the first group of $\sqrt{N}$ neighboring destinations is equal to $(\sqrt{N})! \cdot \sqrt{N}^{\sqrt{N}}$, where $(\sqrt{N})!$ is the number of different permutations of $\sqrt{N}$ MS bits and $\sqrt{N}^{\sqrt{N}}$ is the number of arrangements of $\sqrt{N}$ LS bits of destinations' addresses. The number of arrangements of $\sqrt{N}$ MS bits of destinations in all the other groups is equal to 1, since there is only one way that the $\sqrt{N}$ MS bits can be arranged to be equal to the corresponding MS bits in the first group. Therefore, the number of arrangements for the next group of $\sqrt{N}$ destinations is equal to the number of arrangements of $\sqrt{N}$ LS bits, which is $(\sqrt{N} - 1)^{\sqrt{N}}$. Since there are $\sqrt{N}$ such groups, the total number of maximum-conflict mappings is equal to $(\sqrt{N}!)^{\sqrt{N}+1}$. □

Table 1 presents the fraction of all mappings that produce a maximum number of conflicts as a function of N. Note that this fraction is smaller than the fraction of conflict-free mappings and it decreases rapidly as N grows.

4. Processor-memory mappings for the FFT

Various parallel FFT algorithms have been studied by Pease [6]. Their performance is analyzed in [1]. In this section, we assume that all data items of the one-dimensional FFT accessed by a processor during an iteration are allocated to a single memory module. Therefore, two or more processors cannot access the same memory module during one iteration. We show that if an initial allocation of data to memory modules results in a conflict-free mapping, a conflict-free mapping is produced during all other iterations of the algorithm. Moreover, if an initial allocation results in a maximum-conflict mapping, then mappings for all other iterations of the algorithm are maximum-conflict. Therefore, for the FFT algorithm, an initial allocation of data to memories appears to be very important for controlling the contention within the network.

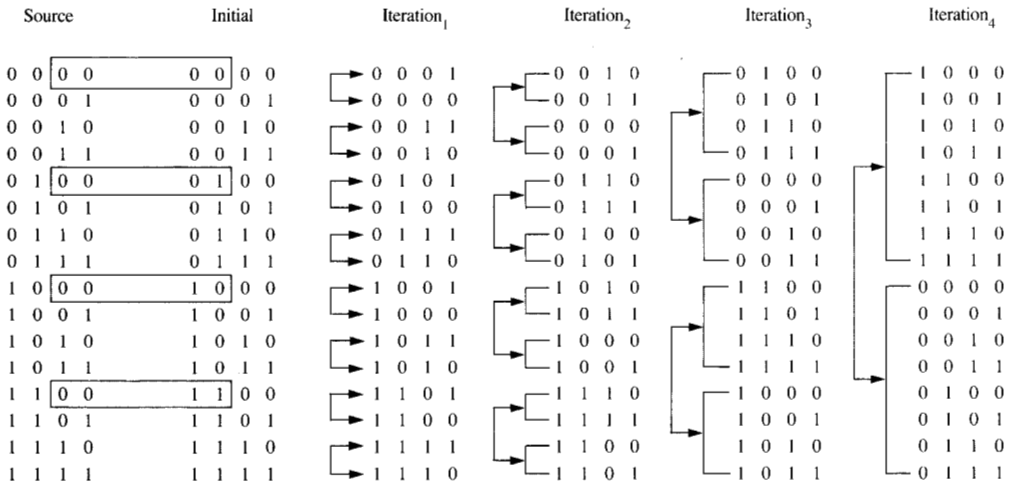

Figure 5

Conflict-free mapping for FFT with $N = 16$ processors.

• *Conflict-free mappings*

In this case, we assume that data items are allocated to memory modules so that N processors can access data from N memories without generating any conflicts within the network. In order to show that no conflicts are generated during any other of $\log N$ iterations of the algorithm, we prove that the property of conflict-free mappings is maintained through all iterations.

Figure 5 illustrates this property using an example with $N = 16$ processors. In this figure, the first column represents binary addresses of 16 processors (sources). The other columns represent binary addresses of 16 memory modules (destinations) accessed by processors during each iteration of the algorithm. Each row contains addresses of the source and all destinations accessed by this particular source during every iteration of the algorithm. The initial mapping shows addresses of those destinations accessed by processors in order to load data before the beginning of the computation.

Claim 4 For the FFT algorithm, if the initial allocation produces no conflicts in the omega network, then no conflicts are produced during any iteration of the algorithm.

Proof We first observe that the mapping from processors to memories during the jth iteration ($j = 1, 2, \cdots, \log N$) of the FFT is equivalent to exchanging consecutive groups of 2^{j-1} neighboring destinations in each group from the initial mapping and then performing a mapping from sources to destinations in the same rows. The groups exchanging their positions are marked in Figure 5 for each iteration of the algorithm. Note that there are $N/2^{j-1}$ such groups in the jth iteration. For example, there are four groups consisting of four elements in the third iteration ($j = 3$). In this example, the second and the fourth group are shifted up and the first and the third group are shifted in the opposite direction.

We now show that for all iterations of the FFT, the sets of $N/2^{\log N-k}$ destinations with distance equal to $2^{\log N-k}$ for network stage $k = 1, 2, \cdots, \log N$ are the same as for the initial mapping.

In order to prove the above statement, for iteration j of the FFT ($j = 1, 2, \cdots, \log N$), we distinguish three different cases:

1. The destinations with distance greater than 2^{j-1} are shifted in the same direction and thus remain at the same distance. Hence, the sets of all destinations with distance greater than 2^{j-1} are the same. From Figure 5 we observe that groups of four elements are exchanged at the third iteration and the destinations with distance equal to 8 move in the same direction. Therefore they remain at the same distance from one another.
2. The consecutive destinations with distance equal to 2^{j-1} are swapped. Therefore, the sets of elements with distance equal to 2^{j-1} are the same as for the initial mapping. The consecutive destinations with distance equal to 4 in the third iteration shown in Figure 5 exchange their positions and thus remain at the same distance.
3. Since the order of destinations within the groups does not change, these destinations also remain at the same

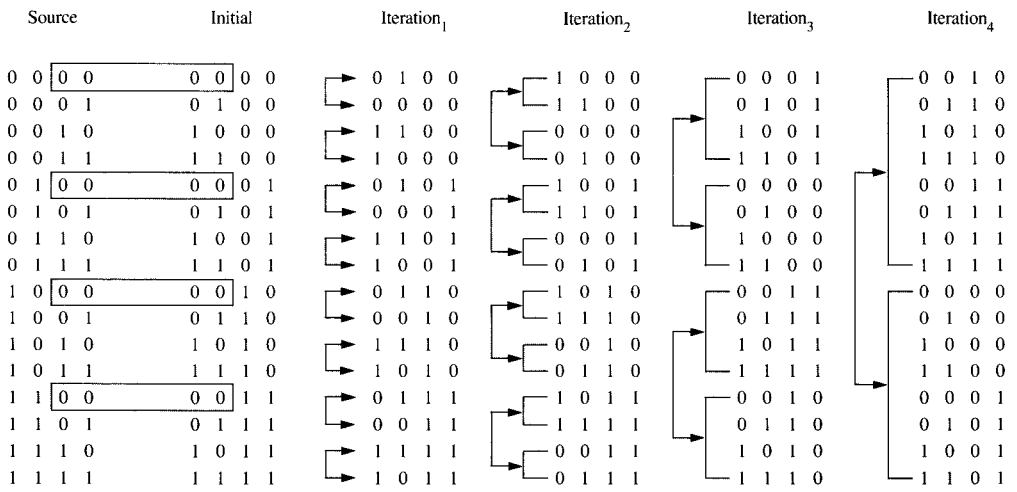

Figure 6

Maximum-conflict mapping for FFT with $N = 16$ processors.

distance from one another. Therefore, the sets of elements with distance less than 2^{j-1} are the same as for the initial mapping. For example, destinations inside the groups of four elements in the third iteration in Figure 5 remain at the same distance from one another.

We conclude that in all three cases the destinations with distance equal to $2^{\log N-k}$, where k is the stage number, remain at the same distance for all iterations of the FFT. Thus, we have proved that each iteration of the FFT results in a conflict-free mapping, provided that the initial mapping is conflict-free. □

• *Maximum-conflict mappings*

As in the previous case, we show that the property of maximum-conflict mappings is maintained during each iteration of the algorithm.

Claim 5 For the FFT algorithm, if the initial allocation results in a maximum-conflict mapping, then all iterations of the algorithm produce a maximum number of conflicts within the omega network.

Proof According to Claim 4, for the jth iteration of the FFT, the sets of elements with distance equal to 2^k ($k = 1, 2, \cdots, \log N$) are the same as for the initial mapping. The same holds for the sets of elements with distance equal to $2^{\lceil (\log N)/2 \rceil}$, which determine a maximum number of conflicts in the network. Therefore, the property of maximum-conflict mappings from Claim 2 is maintained throughout all iterations of the FFT. □

As an example, **Figure 6** shows a maximum-conflict mapping for all iterations of the FFT. From this figure, we observe that if the initial allocation results in a maximum-conflict mapping, then all $\log N$ iterations yield a maximum number of conflicts in the network. In this example, the maximum number of conflicts is equal to 4 and it occurs in the second stage of the network. One of the bottleneck switches in this stage (switch 0) is marked by four windows in this figure. The addresses of the bottleneck switches from this figure are 0, 5, 10, 15 for the initial, third, and fourth iterations; 1, 4, 11, 14 for the first iteration; and 2, 7, 8, 13 for the second iteration.

5. Mappings for the bitonic sort algorithm

In this section we analyze mappings for the bitonic sort algorithm by Batcher [7], which recursively performs sorting on portions of an input array of N data items, starting from smaller and then extending to larger bitonic sequences. This algorithm applies the FFT butterfly network of size N or smaller as a merging structure (**Figure 7**).

Claim 6 For Batcher's bitonic sort algorithm, if an initial allocation results in a conflict-free (maximum-conflict) mapping, then all other iterations of the algorithm are conflict-free (maximum-conflict).

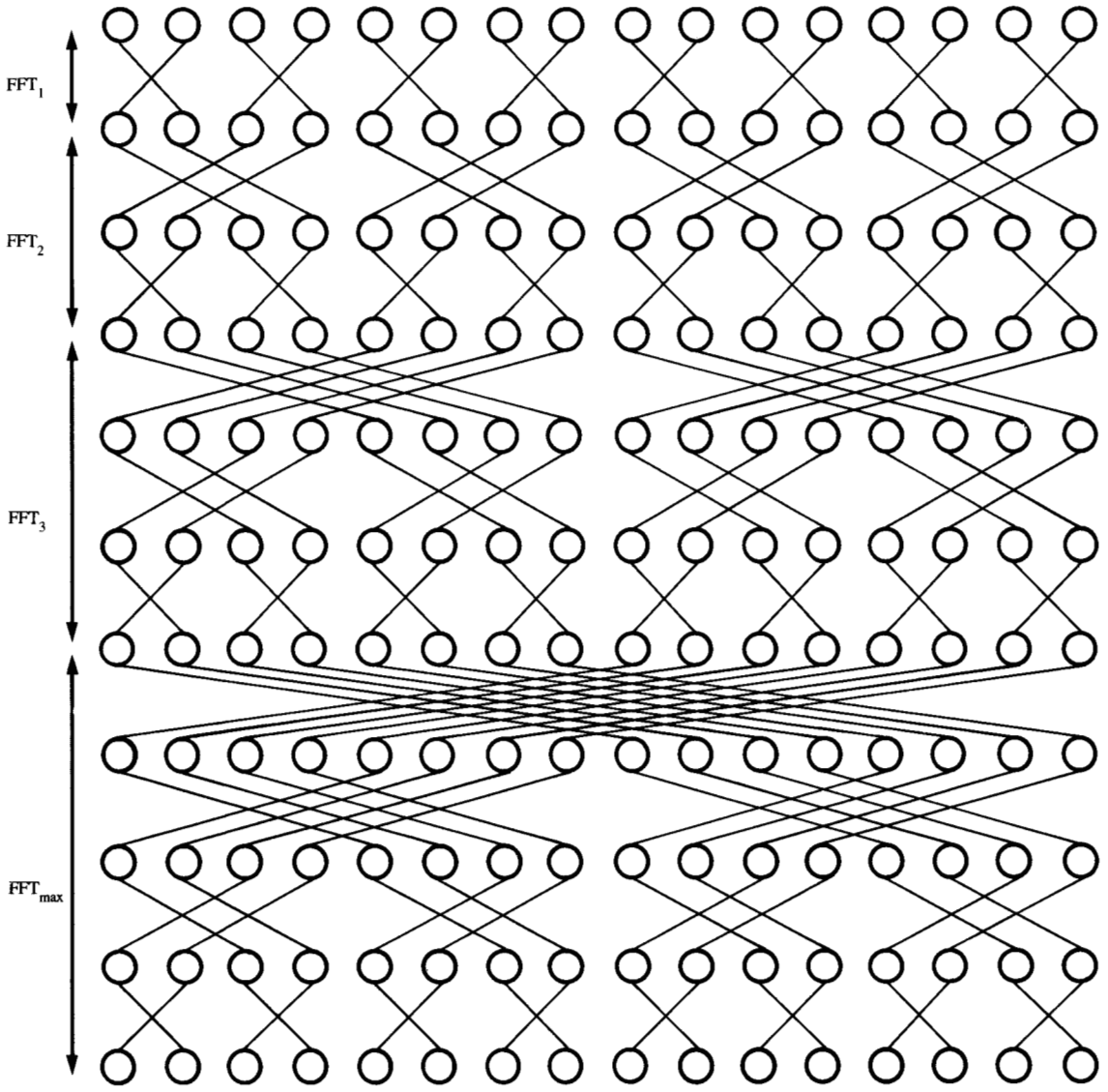

Figure 7

Batcher's bitonic sort algorithm for $N = 16$ data items.

Proof Claims 4 and 5 state that the property of the initial mapping is maintained through all $\log N$ iterations of the FFT. The bitonic sort algorithm applies the FFT graph of size N and all the other FFT graphs of smaller sizes. Since all these smaller graphs are embedded in the graph of size N, the property of the initial mapping is also extended to them. Therefore, the bitonic sort algorithm preserves the property of the initial mapping. □

For example, in Figure 7 the smaller graphs FFT_1 with $\log N - 3$ iterations, FFT_2 with $\log N - 2$ iterations, and FFT_3 with $\log N - 1$ iterations are embedded in the full-size graph FFT_{max} with $\log N$ iterations.

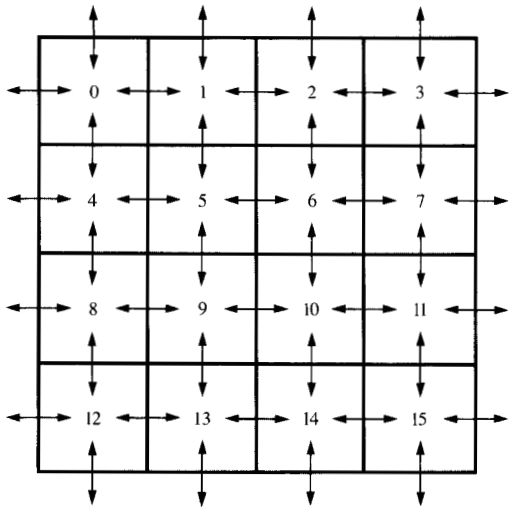

Figure 8

Example of a conflict-free allocation for a 4×4 grid.

6. Mappings for nearest-neighbor communication

In this section, we analyze mappings for the d-dimensional grid computations in which communication is required with $2d$ neighbors. We consider processor-memory mappings in which all data items are loaded into global memories before the computation is begun. During the computation, each data access requires traversing the network.

We assume that a computation is performed at each point of a $N^{1/d} \times N^{1/d} \times \cdots \times N^{1/d}$ d-dimensional grid of data. For example, **Figure 8** shows a 4×4 grid of data which is mapped into a system consisting of $N = 16$ processors. We assume that the grid *wraps around* such that, in this example, the first row is adjacent to the $\sqrt{N}$th row and the first column is adjacent to the $\sqrt{N}$th column. During each iteration, a processor references one data item (initial mapping) and this item's upper, lower, right, and left neighbors. The addresses of the memory modules containing data from the grid for a conflict-free allocation are indicated by the numbers associated with each grid point. **Figure 9** shows conflict-free mappings for four-neighbor communication with $N = 16$ processors and a 4×4 grid of data.

Claim 7 If data items of a d-dimensional grid are allocated to N memories ($N = 2^n$) such that the initial mapping is conflict-free (maximum-conflict), then the communication with $2d$ neighbors of each grid item, where n/d is an integer, is also conflict-free (maximum-conflict).

Proof The mappings required by the network include a rotation by $N^{(d-i)/d}$ positions up and down, where n/d is an integer, in the groups of $N^{(d-i+1)/d}$ destinations for all

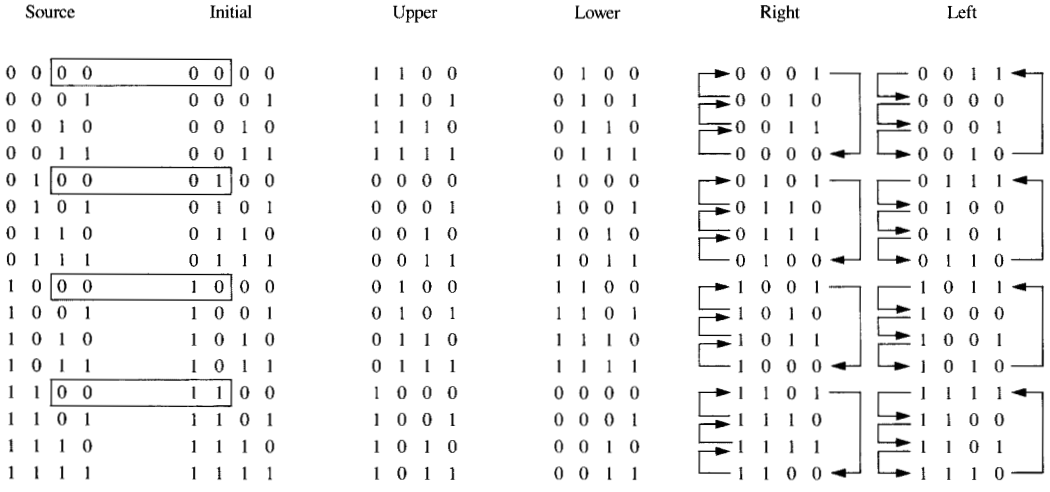

Figure 9

Conflict-free mapping for four-neighbor communication and $N = 16$ processors.

neighbors $i = 1, 2, \cdots, d$. For example, in Figure 9 a request for data from the upper (lower) neighbor is equivalent to the rotation of destination addresses from the initial mapping by $\sqrt{N}$ positions down (up), while a request for data from the right (left) neighbor is equivalent to the rotation by one position up (down) in the groups of $\sqrt{N}$ neighboring destinations. The third (fourth) column in Figure 9 is obtained by a rotation by four positions down (up) of destination addresses from the initial mapping. The fifth (sixth) column from this figure is obtained by rotating the destinations by one position up (down) in the groups of four destinations.

All of these mappings are conflict-free (maximum-conflict) provided that the initial mapping is conflict-free (maximum-conflict). This is true because by such rotations in the groups of $2^{n(d-i+1)/d}$ destinations where n/d is an integer, the destinations remain at the same distance from one another. Therefore, the properties of an initial mapping from Claims 1 and 2 are preserved. □

7. Bit-permute mappings

In this section, we show that the omega network cannot produce without conflicts some of the common bit-permute mappings of an identity mapping. Specifically, it cannot produce the perfect-shuffle and the bit-reversal permutations.

- *Perfect-shuffle mapping*

The perfect-shuffle mapping is obtained by applying the perfect-shuffle permutation to the destinations, such that the destination i is replaced by the destination $2i \bmod (N-1)$ for all $i = 0, 1, \cdots, N-1$. This mapping is equivalent to the left rotation of the destination address bits from the initial mapping. If the initial mapping is an identity mapping, then the number of MS bits that are identical for destinations with distance equal to 2^k, where $k = 1, 2, \cdots, \log N - 1$, is increased two times after the perfect-shuffle permutation.

Figure 10(a) illustrates the perfect shuffle of an identity mapping for $N = 16$ processors. The second column shows the initial mapping, which is an identity mapping, while the third column presents the perfect shuffle of destinations from the initial mapping. In this example, the number of conflicts increases from 0 to 2 after the perfect-shuffle permutation. The bottleneck switches are determined by sources and destinations with distance equal to 8.

We now show that data items can be accessed from memories such that if an initial mapping is conflict-free, then the perfect shuffle of such mapping is also conflict-free. **Figure 10(b)** shows the allocation in which data items are accessed from memories in a skewed manner, resulting in both the initial mapping and the perfect-shuffle mapping being conflict-free. This mapping keeps the first $N/2$ destinations in the same order as for the identity mapping, while the second $N/2$ destinations are skewed such that shifting their addresses one position to the left results in different switch addresses in each network stage.

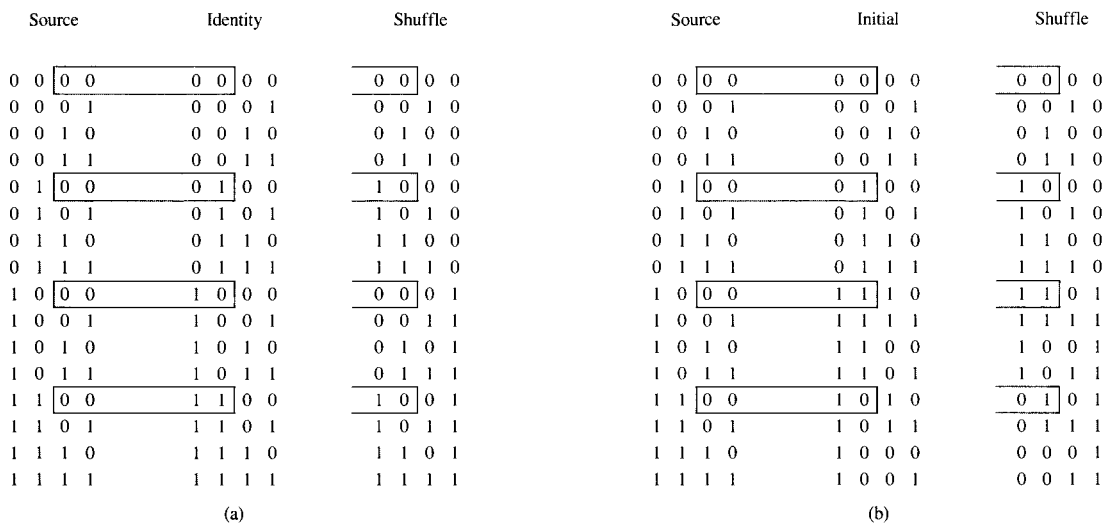

Figure 10

Perfect shuffle of mappings for $N = 16$ processors: (a) identity mapping, (b) skewed conflict-free mapping.

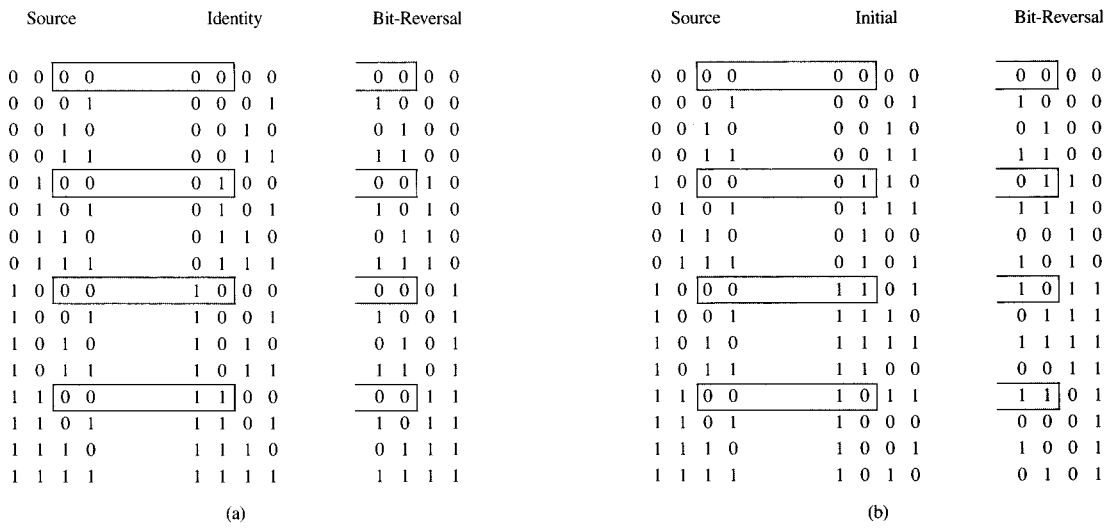

Figure 11

Bit reversal of mappings for $N = 16$ processors: (a) identity mapping, (b) skewed conflict-free mapping.

Note also that the property of a maximum-conflict mapping is not preserved by the perfect-shuffle permutation. This is true since the $\sqrt{N}$ LS bits are different for all destinations at a distance equal to $\sqrt{N}$, thus preventing the same switch address from being generated after the perfect-shuffle permutation.

• *Bit-reversal mapping*

The bit-reversal mapping is obtained by applying the bit-reversal permutation to the destinations, such that the new destination addresses are obtained by reversing bits from the old addresses. After the bit-reversal mapping, the $(\log N)/2$ MS bits of the destination addresses exchange their positions with the $(\log N)/2$ LS bits in reverse order. For the identity mapping, which is conflict-free, the $(\log N)/2$ MS bits of all destinations with distance equal to $\sqrt{N}$ are different, and thus the $(\log N)/2$ LS bits of these destinations must be the same. Hence, with the bit reversal, the identity mapping is transformed into a maximum-conflict mapping.

Figure 11(a) illustrates the bit reversal of an identity mapping for $N = 16$ processors. The third column represents the bit reversal of destinations from the second column (identity mapping). In this example, the number of conflicts is increased from 0 to 4 after the bit reversal of a conflict-free mapping. The set of destinations that determine one of four switches with four conflicts is indicated in the figure by four windows. Note that conflicts appear in the second network stage, which is in this case the bottleneck stage.

Figure 11(b) shows a conflict-free mapping for which the bit-reversal permutation is also conflict-free.

Note further that the property of a maximum-conflict mapping cannot be preserved by the bit reversal, since for the destinations at distance $\sqrt{N}$ the $\sqrt{N}$ LS bits are different and thus cannot result in the same switch address after the bit reversal.

8. Conclusions

The omega network belongs to a class of multistage networks for use in both a dataflow architecture as an interconnection structure among processors, and a shared-memory multiprocessor architecture where it connects processors to global memories. In this paper, we have analyzed several synchronous structured parallel algorithms which apply a static allocation of data to memories and where all requests from N processors arrive at the network simultaneously. The common property of the algorithms analyzed here is that they repeat the same computation in cycles or iterations, and that the communication pattern is well defined during each iteration. We showed that different allocations of data to memory modules cause the contention within the network to vary significantly. For conflict-free allocations, no conflicts are produced within the network and the communication delay grows as $O(\log N)$ if network stages are not pipelined, or as $O(1)$ if network stages can be pipelined. For maximum-conflict allocations, a maximum number of conflicts are produced in the network and the

communication delay is increased from $O(\log N)$ to $O(\log N + \sqrt{N})$ if network stages are not pipelined, or from $O(1)$ to $O(\sqrt{N})$ if network stages can be pipelined. Therefore, by identifying and applying conflict-free allocations, the communication delay can be improved by a factor of $O(\sqrt{N})$ during each iteration of the algorithm.

One surprising finding of this study for the FFT and bitonic sort algorithms is that if an initial allocation results in a conflict-free mapping, then all other iterations of the algorithm produce a conflict-free mapping. Similarly, if an initial allocation results in a maximum-conflict mapping, then all other iterations of the algorithm produce a maximum-conflict mapping. It would be interesting to determine whether the results obtained for the FFT and bitonic sort algorithm can be extended to other structured iterative applications with similar characteristics. The butterfly graph used to implement the FFT exhibits a global communication property in that each input affects each output through $\log N$ stages of the graph. We conjecture that the results obtained here can be applied for other algorithms with the same communication property. Consider, for example, prefix-computation algorithms, which are related to linear recurrences and binary addition. The computation graph for a prefix computation is similar to that of the FFT network. It applies a uniform shift of outputs at each stage of the algorithm, and hence it can be realized by the omega network, provided that an initial allocation is conflict-free.

The second group of mappings analyzed in this paper includes algorithms where communication is required with $2d$ nearest neighbors. These mappings share a local communication property, since input data items affect only a limited number of outputs. For these algorithms, there is even more freedom for avoiding network contention by applying a conflict-free allocation. Other problems with similar properties include tree computations, where each node in the tree affects only a limited number of descendant nodes, and matrix operations such as matrix multiplication, where the computation of each output matrix element requires access to one row of the first matrix and one column of the second matrix. We believe that for these problems, careful allocation of data to memories can reduce network contention considerably.

It would be also interesting to investigate whether the same results can be obtained for very large problems where the problem size exceeds the number of processors available. The problems analyzed here are reducible, so that the graph of smaller size exhibits the same communication properties as the original graph.

The results obtained here indicate that the initial allocation of data to memory modules is significant for determining network contention during all other iterations of several algorithms analyzed here. Since many applications intended for parallel execution require a large number of iterations, we believe that it is worthwhile to investigate methods for allocating data to memories as one of the strategies for avoiding network bottlenecks and thereby significantly improving the performance of multiple-processor systems.

Acknowledgments

The author would like to thank Harold S. Stone for his numerous comments and suggestions during this research, and David Carlson for his careful reading of early versions of this paper and valuable suggestions. This research was supported in part by the National Science Foundation under Grant No. MCS-7805298 and by the IBM Corporation under Contract No. 462914.

References

1. (a) Ž. Cvetanović, "Performance Analysis of Multiple-Processor Systems," Ph.D. dissertation, Department of Electrical and Computer Engineering, University of Massachusetts, Amherst, MA, May 1986. (b) Ž. Cvetanović, "The Effects of Problem Partitioning, Allocation, and Granularity on the Performance of Multiple-Processor Systems," *IEEE Trans. Computers* **C-36,** No. 4, 421–432 (April 1987).
2. D. H. Lawrie, "Access and Alignment of Data in an Array Processor," *IEEE Trans. Computers* **C-24,** No. 12, 175–189 (December 1975).
3. H. S. Stone, "Parallel Processing with the Perfect Shuffle," *IEEE Trans. Computers* **C-20,** No. 2, 153–161 (February 1971).
4. T. Lang, "Interconnection Between Processors and Memory Modules Using the Shuffle-Exchange Network," *IEEE Trans. Computers* **C-25,** No. 5, 496–503 (May 1976).
5. P.-Y. Chen, D. H. Lawrie, P.-C. Yew, and D. A. Padua, "Interconnection Networks Using Shuffles," *Computer* **14,** No. 12, 55–64 (December 1981).
6. M. C. Pease, "An Adaptation of the Fast Fourier Transform for Parallel Processing," *J. ACM* **15,** 252–264 (April 1968).
7. K. E. Batcher, "Sorting Networks and Their Applications," *1968 Spring Joint Computer Conference, AFIPS Proc.* **32,** 307–314 (1968).

Received April 21, 1986; accepted for publication January 23, 1987

Žarka Cvetanović *Digital Equipment Corporation, 85 Swanson Road, Boxborough, Massachusetts 01719.* Dr. Cvetanović received a Ph.D. in computer and electrical engineering from the University of Massachusetts at Amherst in 1986. She is currently a principal engineer at the Digital Equipment Corporation. This paper was conceived and written during her summer 1985 appointment at the IBM Thomas J. Watson Research Center in Yorktown Heights, New York. Her research interests are in the area of parallel algorithms and performance analysis of multiple-processor systems.