Performing BMMC Permutations in Two Passes Through the Expanded Delta Network and MasPar MP-2
(Extended Abstract)

Leonard F. Wisniewski*
Thomas H. Cormen†
Thomas Sundquist‡

Abstract

One basic operation that we expect multiprocessor interconnection networks to perform efficiently is permuting. When messages are dropped due to collisions in the network, we often measure the efficiency of a permuting method on the network by counting the number of times data must pass through the network to perform the desired permutation. There are sometimes advantages to studying restricted classes of permutations on specific networks: they may be performed in fewer passes than general permutations, or even if they take the same number of passes, determining how to route them is simpler.

This paper examines routing of BMMC (bit-matrix-multiply/complement) permutations on two types of multistage interconnection networks: the expanded delta network and the global router of the MasPar MP-2. BMMC permutations are an important class of permutations that has been well-studied on various multistage networks. The class of BMMC permutations includes as subclasses Gray-code and inverse Gray-code permutations and the entire subclass of bit-permute/complement (BPC) permutations, which in turn includes matrix transpose (with power-of-2 dimensions), bit reversal, vector reversal, hypercube, and matrix reblocking permutations.

There are four results in this paper. First, we use linear-algebraic techniques to derive an algorithm to perform any BMMC permutation in at most two passes on the expanded delta network. Second, we use linear-algebraic techniques to derive an algorithm to perform any BMMC permutation in at most two passes on the global router of the MasPar MP-2. Third, we use linear-algebraic and combinatorial analysis to determine the distribution of all BMMC permutations when routed naively through the MP-2 global router and show that most, but not all, BMMC permutations require only one or two passes anyway. We can apply our two-pass algorithms in those cases when naive routing requires more than two passes. Fourth, we present experimental evidence to support our analysis.

*Thinking Machines Corporation. Research performed while at Dartmouth College Department of Computer Science. Supported in part by a Dartmouth Fellowship and the National Science Foundation under Grant CCR-9308667. Contact author's email address: `lenbo@think.com`.

†Department of Computer Science, Dartmouth College. Supported in part by funds from Dartmouth College and in part by the National Science Foundation under Grant CCR-9308667.

‡Department of Mathematics, Dartmouth College. Supported in part by funds from Dartmouth College.

1 Introduction

When performing permutations on a distributed-memory multiprocessor, interprocessor communication can require a significant portion of the overall execution time. In a distributed-memory multiprocessor, we evenly partition M records of data across P processors, i.e., M/P records per processor. We connect the processors by an interconnection network. To permute data, we must perform interprocessor communication to send each message (containing a datum to move) across the interconnection network from its current processor to the processor with its target location. For some permutations, we can utilize the structure of the permutation to take advantage of the structure of the interconnection network to efficiently perform the interprocessor communication.

Multistage networks comprise one such class of structured interconnection networks for distributed-memory multiprocessors. A multistage network consists of alternating stages of switches and wires. The switches are typically crossbars connected by wires, usually in a pattern based on the perfect shuffle [Sto71].

In this paper, we use a linear-algebraic framework to develop algorithms for efficiently performing the class of BMMC (bit-matrix-multiply/complement) permutations on multistage networks. We assume that each input and output is the source and destination, respectively, of exactly one message for each permutation and that the permutation is a BMMC permutation. This paper includes the following results:

- an algorithm to perform any BMMC permutation in two passes on the expanded delta network, or EDN,
- an algorithm to perform any BMMC permutation on the global router of the MasPar MP-2 in two passes,
- analysis of the BMMC characteristic matrix to derive a probability distribution for the number of passes through the global router of the MasPar MP-2 to perform an arbitrary BMMC permutation when using naive routing,

- experimental data on several sizes of the MasPar MP-2 to confirm the probability distribution for BMMC permutations that we derived.

The probability distribution shows that we can use naive routing to perform most BMMC permutations in one or two passes through the MasPar MP-2 global router. In those cases when naive routing requires more than two passes, we can apply our two-pass algorithm.

The expanded delta network is a scalable network, and the two-pass algorithm for performing BMMC permutations works on EDNs of any size. On the other hand, the largest MasPar MP-2 has 16K processors and, to the best of our knowledge, the MP-2 global router cannot exceed this size. To further complicate the picture, the largest MP-2 global router is an EDN, but all smaller MP-2 global routers are not. Therefore, we developed two routing methods for BMMC permutations: one for the scalable EDN, and one for the more limited MasPar MP-2.

Previous work on multistage networks

Many multistage interconnection networks have been proposed, such as the Ω network [Law75], the indirect binary n-cube [Pea77], the Clos network [Clo53], the Benes network [Ben65], the delta network [Pat81], and the expanded delta network (EDN) [KS83]. The switches are typically crossbars connected by wires, usually in a pattern based on the perfect shuffle [Sto71]. The above networks are equivalent, topologically and functionally [Pea77, Sie77].

Researchers have developed efficient algorithms on these multistage networks. A *pass* through the network is a rearrangement in which some subset of the data simultaneously moves through the network to a destination. Since any arbitrary permutation can be routed through a multistage network if multiple passes are allowed, we say that the above multistage networks are *universal.* Waksman [Wak68] and Kautz [Kau68] describe how to perform an arbitrary permutation of N elements using basic 2-permuters. Parker [Par80] shows a one-pass algorithm for an arbitrary permutation on the three-stage Benes network and also presents a three-pass algorithm on the Ω network. Scherson and Subramanian [SS93] give an off-line algorithm to route an arbitrary permutation in three passes on a restricted-access EDN. In a restricted-access EDN, the number M

of elements may be greater than the number P of processors, so that there are multiple records per processor, but each network port routes at most one record per pass.

Several algorithms appear in the literature for routing certain classes of permutations in fewer than three passes. For example, several papers have presented algorithms for routing BMMC (bit-matrix-multiply/complement) permutations, which we define below, and its subclass of BPC (bit-permute/complement) permutations. Pease [Pea77] shows how to perform BMMC permutations in two passes on the indirect binary n-cube. Keohane and Stearns [KS88] give an algorithm for performing linear permutations (BMMC permutations without the complement) in two passes on the Ω network. Yew and Lawrie [YL81] present a two-pass algorithm for BPC permutations on the Ω network, and Nassimi and Sahni [NS81] give an algorithm for BPC permutations on a Benes network. Shen [She95] shows how to perform any BPC permutation on a k-extra-stage Ω network in a minimum number of passes. Scherson and Subramanian's general permutation algorithm [SS93] only requires two passes on a certain class of EDNs, but it has a high cost of decomposing the permutation.

Cormen and Bruhl [Bru94, CB95] study the performance of several algorithms for BMMC permutations on the restricted-access MasPar MP-2 global router. Their algorithms adapt the block BMMC algorithms for parallel disk systems with a block size of 1 from [Cor92, Cor93] to perform BMMC permutations on the global router when the number of messages in the data set exceeds the number of inputs and outputs. They present algorithms that guarantee that each processor or cluster of processors sends and receives one message at a time. Thus, with the methods of Cormen and Bruhl, we are able to abstract away the notion of virtual processors and concentrate on performing BMMC permutations of data sets with exactly the same number of inputs and outputs as a given EDN or MasPar MP-2. That is, their algorithms determine in which order to admit messages into the network, but they do not attempt to control how messages are routed once they enter. The present paper, on the other hand, is concerned with routing messages only after they have entered the network.

Outline of this paper

Section 2 defines the class of BMMC permutations and presents some simple decomposition properties. In Section 3, we define the class of expanded delta networks and present our general approach to applying the linear-algebraic framework to multistage networks. We apply the framework to EDNs and derive a two-pass algorithm for performing BMMC permutations. Section 4 again applies the linear-algebraic framework to determine how to perform BMMC permutations in two passes through the global router of the MasPar MP-2. In Section 5, we derive a probability distribution for the number of passes through the MasPar MP-2 when we use naive routing and, from that, we compute the expected number of passes. We show that those values are functions of the rank of a specific submatrix within the characteristic matrix for the BMMC permutation. We also present empirical evidence to verify our probability distribution on a range of sizes for the MasPar MP-2. Section 6 concludes.

2 BMMC permutations

This section defines BMMC permutations and presents some of their simple decomposition properties.

The most general class of bit-defined permutations that we consider here is the class of *bit-matrix-multiply/complement*, or *BMMC*, permutations.[1] Throughout the remainder of this paper, for convenience, we use the notation $m = \lg M$. A BMMC permutation of M records is defined by an $m \times m$ *characteristic matrix* $A = (a_{ij})$ whose entries are drawn from $\{0, 1\}$ and is nonsingular (i.e., invertible) over $GF(2)$,[2] and a *complement vector* $c = (c_0, c_1, \ldots, c_{m-1})$ of length m. Treating a *source address* x as an m-bit vector, we perform matrix-vector multiplication over $GF(2)$ and then form the corresponding m-bit *target address* y by complementing some subset of the resulting

[1] Edelman, Heller, and Johnsson [EHJ94] call BMMC permutations *affine transformations* or, if there is no complementing, *linear transformations*.

[2] Matrix multiplication over $GF(2)$ is like standard matrix multiplication over the reals but with all arithmetic performed modulo 2. Equivalently, multiplication is replaced by logical-and, and addition is replaced by exclusive-or.

bits: $y = A\,x \oplus c$, or

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_{m-1} \end{bmatrix} = \begin{bmatrix} a_{00} & a_{01} & a_{02} & \cdots & a_{0,m-1} \\ a_{10} & a_{11} & a_{12} & \cdots & a_{1,m-1} \\ a_{20} & a_{21} & a_{22} & \cdots & a_{2,m-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m-1,0} & a_{m-1,1} & a_{m-1,2} & \cdots & a_{m-1,m-1} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_{m-1} \end{bmatrix} \oplus \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{m-1} \end{bmatrix} .$$

Our convention is to place least significant bits on the top and left.

Although not all permutations are BMMC, many permutations often encountered in practice are. BPC permutations are BMMC permutations in which the characteristic matrix is a permutation matrix, containing one 1 in each row and in each column. Matrix transpose (with power-of-2 dimensions), bit reversal (used in performing FFTs), vector reversal, and matrix reblocking are all BPC, and hence BMMC, permutations. Gray-code permutations, inverse Gray-code permuations, and permutations used by fast cosine transforms [MW95] are BMMC (but not BPC). BMMC permutations are closed under composition so that, for example, the composition of bit-reversal and Gray-code permutations is BMMC.

We shall generally focus on the matrix-multiplication portion of BMMC permutations rather than on the complement vector. A key technique that we use is factoring a characteristic matrix into multiple matrix factors, each of which is nonsingular and of a desired form. If we factor a characteristic matrix A as $A = A^{(k)}\,A^{(k-1)}\,A^{(k-2)}\cdots A^{(1)}$, then we can perform the BMMC permutation characterized by A by performing, in order, the BMMC permutations characterized by $A^{(1)}, A^{(2)}, \ldots, A^{(k)}$. That is, we perform the permutations characterized by the factors of a matrix from right to left.

3 A two-pass BMMC algorithm for expanded delta networks

In this section, we define the expanded delta network and its associated terminology. After giving a brief overview of the application of the linear-algebraic methodology to multistage networks, we present our two-pass algorithm for performing BMMC permutations on the EDN. Appendix A shows how to trace the permutations that occur at each stage of switches and wires to verify that each pass through the EDN performs the permutation specified by the algorithm.

Expanded delta networks

We construct an *expanded delta network (EDN)* [KS83] with *wires* and *switches*. There are two kinds of wires in an EDN: thin wires and thick wires. A *thin wire* carries a single message, whereas each *thick wire* can carry up to K messages. There are three types of switches: thin-to-thick converters, thick-to-thin converters, and hyperbars. A *thin-to-thick converter* copies up to K incoming messages, each on a thin wire, onto a thick wire. A *thick-to-thin converter* splits K messages on a thick wire onto K separate thin-wire outputs. Each incoming message to a thick-to-thin converter specifies a connection request, which is a number between 0 and $K-1$, for the thin-wire output on which to route the message. Each *hyperbar* has S thick-wire inputs and S thick-wire outputs. Each incoming message to a hyperbar contains a connection request, which is a number between 0 and $S-1$, to specify on which thick-wire output to route the message. If more than K incoming messages request a particular thick-wire output, then the hyperbar routes only K of those requests to outputs.

In an EDN, such as the one shown in Figure 1, there are M thin-wire inputs which enter M/K thin-to-thick converters. Similarly, there are M thin-wire outputs which emerge from M/K thick-to-thin converters. Between the thin-to-thick and thick-to-thin converters, data pass through $\log_S(M/K)$ stages of the network. Each stage consists of M/KS hyberbars preceded by an S-way shuffle of the thick wires. If we number the thick wires from 0 to $M/K-1$, an S-way shuffle rotates the binary representation of the index for each thick wire $\lg S$ positions toward the more significant positions. In this section, we shall assume that M, S, and K are powers of 2, and we use the following notation for convenience: $m = \lg M$, $s = \lg S$, and $k = \lg K$. Since there are $\log_S(M/K)$ stages in the network and the number of stages is an integer, we assume that s evenly divides $m-k$.

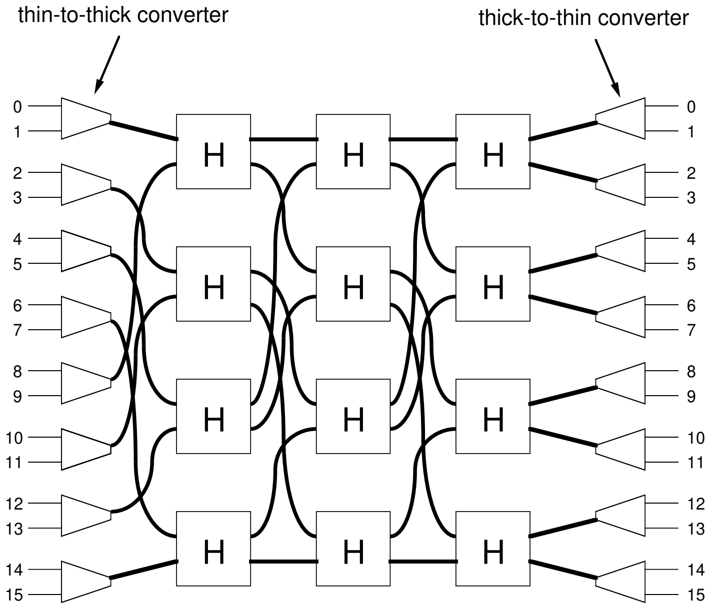

Figure 1: An expanded delta network with $M = 16$, $S = 2$, and $K = 2$. The numbers along the left and right signify the addresses of the input and output messages, respectively. Thin lines represent thin wires and thick lines represent thick wires. Exactly S thick wires enter and exit each hyperbar (squares containing an "H").

Information passes through an EDN as follows. A *path* is the set of wires traversed by a message from its thin-wire input to its thin-wire output. Scherson and Subramanian [SS93] state that this path is unique for each input. If we have a set of processors $\mathcal{P} = \{\mathcal{P}_0, \mathcal{P}_1, \ldots, \mathcal{P}_{M-1}\}$, we connect each input wire i and output wire i to processor $\mathcal{P}_i$. We represent a *message* as an ordered pair (i, j), originating from processor i and destined for processor j. A *traffic pattern* Γ is a set of messages, i.e., a relation over $\mathcal{P}$. We say that a traffic pattern Γ is *EDN passable*, or *collision-free*, if

- for each thin wire w in the EDN, there is at most one message $(\mathcal{P}_i, \mathcal{P}_j) \in \Gamma$ such that w lies on the unique path from input port i to output port j, and
- for each thick wire w in the EDN, there are at most K messages $(\mathcal{P}_i, \mathcal{P}_j) \in \Gamma$ such that w lies on the unique path from input port i to output port j.

A permutation is a traffic pattern in which each processor is the source of exactly one message and the destination of exactly one message.

Decomposing an arbitrary characteristic matrix into two factors

Here, we show how to decompose the characteristic matrix for any BMMC permutation into two factors, each of which has a form that characterizes a permutation that can be performed in one pass through an EDN. The rightmost factor defines a permutation of the messages to *intermediate locations*, as in the style of the Valiant-Brebner algorithm for hypercube routing [VB81]. (Our method is deterministic, whereas the Valiant-Brebner method is randomized.) After permuting the messages to the intermediate locations, we permute the messages to their final locations according to the factor on the left.

Both factors have an important property. We say that a matrix is *all-trailing nonsingular* if all of its trailing square submatrices are nonsingular. (A *trailing submatrix* is one that consists of the bottommost columns and rightmost rows.) Both factors are designed to be all-trailing nonsingular. In Appendix A, we prove that any BMMC permutation whose characteristic matrix is all-trailing nonsingular is EDN passable. To summarize the proof here, we couch each stage of hyperbar switches and wires as a BMMC permutation. As data progress through the stages within the EDN, it undergoes a composition of BMMC permutations. In Appendix A, we show that an all-trailing nonsingular matrix is expressible as the product of matrices corresponding to the stages of the EDN, and hence the permutation is EDN passable.

Performing the factoring is relatively easy; we provide a brief description here. We produce a matrix $\widehat{A}$ that is all-trailing nonsingular by adding rows of the characteristic matrix A. We produce the rows of $\widehat{A}$ from bottom to top, where initially $\widehat{A} = A$. Each row may have rows above added in (with all additions over $GF(2)$). We account for each row addition in a *row-addition matrix* Q, which is an identity matrix augmented with a 1 in each (i, j) entry whenever we add row j of $\widehat{A}$ to row i. When we are done reducing, we have that $\widehat{A} = Q\,A$. Because we only add lower-indexed rows to higher-indexed rows, the matrix Q is unit lower-triangular. Hence, Q^{-1} exists and is also unit lower-triangular. Multiplying both sides by Q^{-1}, we have factored $A = Q^{-1}\widehat{A}$. Since each trailing square submatrix of a unit lower-triangular matrix is also unit lower-triangular, the factor Q^{-1} is all-trailing nonsingular.

We produce $\widehat{A}$ by ensuring that the trailing $l \times l$ submatrix is nonsingular, for $l = 1, 2, \ldots, m$, in order. By induction, assume that the trailing $(l-1)\times(l-1)$ submatrix is nonsingular, where a 0×0 matrix is nonsingular by default. Because $\widehat{A}$ is produced by adding upper rows of a nonsingular matrix to lower rows, it is nonsingular at all times. Therefore, the l rightmost columns of $\widehat{A}$ have row rank l. Moreover, because the trailing $(l-1)\times(l-1)$ submatrix is nonsingular, the trailing $(l-1)\times l$ submatrix has row rank $l-1$. If the trailing $l \times l$ submatrix is already nonsingular, we don't need to do anything. Otherwise, it is singular because its top row is dependent on the bottom $l-1$ rows. Since the l rightmost columns have row rank l, there must be a row of $\widehat{A}$ above the trailing $l \times l$ submatrix that we can add into that top submatrix row to make the submatrix nonsingular. We do so, updating the entire row of $\widehat{A}$ and the row-addition matrix Q accordingly.

Keohane and Stearns [KS88] derived a two-pass algorithm for performing linear permutations on the Ω network, but they did not use the linear-algebraic technique presented here and in Appendix A. Since an EDN allows more flexibility than an Ω network, one could apply their algorithm directly to an EDN, restricting the data movement within each hyperbar and restricting the placement of data on each thick wire. Our linear-algebraic technique provides a clearer representation of the component permutations performed at each stage.

4 Two-pass BMMC algorithm for MasPar MP-2

In this section, we derive a two-pass algorithm on the global router of any size of the MasPar MP-2, up to the full 16K-processor version. The global router of the MasPar MP-2 with 16K processors is an expanded delta network, but the smaller sizes of the MasPar MP-2 are not. We first describe the network architecture of the MasPar MP-2. Then we define the restricted characteristic matrix forms for each stage of the MasPar MP-2. Appendix B presents our two-pass algorithm for performing

BMMC permutations on the MasPar MP-2.

Global router of the MasPar MP-2

Again, we have M inputs and M outputs to our network. Between the M thin-wire inputs and the M thin-wire outputs, the *global router* of the MasPar MP-2 includes three stages of switches and two stages of wires. We illustrate a MasPar MP-2 global router with $M = 128$ inputs and outputs in Figure 2. The first stage consists of $M/64$ hyperbars, each with 64 thin-wire inputs and 16 thick-wire outputs with 4 messages each, i.e., $K = 4$. Each input can request a specific thick wire, but has no control over its location within the thick wire. The second stage consists of 16 hyperbars, each of which has $M/64$ thick-wire inputs and outputs. The third stage uses $M/4$ 4×4 crossbar switches. We arrange the thick wires between the first and second stages of hyperbars in a perfect shuffle permutation. We arrange the thick wires between the second stage of hyperbars and the third stage of crossbars in the inverse of the perfect shuffle permutation. The size of a MasPar MP-2 ranges from a 1K processor element (PE) model to a 16K PE model. Because each group of 16 PEs shares a connection to the global router, $M = 64$ on the 1K PE model and $M = 1024$ on the 16K PE model.

Matrix factors corresponding to each stage

We represent the three stages of switches by the characteristic matrix forms L, C, and R, corresponding to the left, center, and right stages, respectively, in Figure 2. To determine a specific characteristic matrix for each stage, we must obey the restrictions on the characteristic matrix for each stage.

We first characterize the left stage of $M/64$ hyperbars. If we let $m = \lg M$ and we number the hyperbars from 0 to $M/64 - 1$, the most significant $m - 6$ bits of a source address indicate the *hyperbar number* for a particular message. Since each message remains on a particular hyperbar for the entire switching stage, the characteristic matrix corresponding to the left stage cannot change the hyperbar number for a message. Thus, we characterize the left stage by a nonsingular $m \times m$

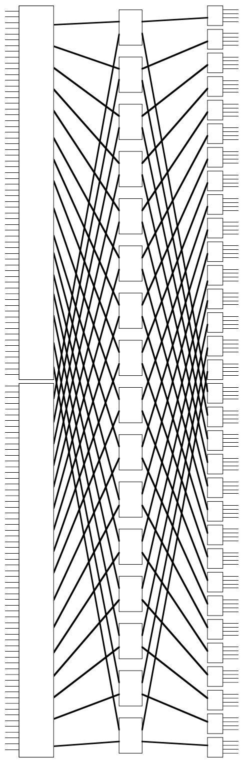

Figure 2: The MasPar MP-2 global router when $M = 128$, $K = 4$, and $S = 2$. The leftmost stage has 2 switches, each of which has 64 thin-wire inputs and 16 thick-wire outputs. The center stage consists of 16 hyperbars, each of which has 2 thick-wire inputs and outputs. The rightmost stage consists of 32 4×4 hyperbars with 1 thick-wire input and 4 thin-wire outputs. The left wiring stage is a 2-way shuffle of thick wires and the rightmost stage is the 2-way unshuffle.

matrix L with a trailing $(m-6) \times (m-6)$ identity matrix and all zeros in the lower left $(m-6) \times 6$ submatrix:

$$L \;=\; \begin{array}{cc} \begin{array}{cc} 6 & \quad m-6 \end{array} & \\ \left[\begin{array}{c|c} \text{nonsingular} & \text{arbitrary} \\ \hline 0 & I \end{array}\right] & \begin{array}{c} 6 \\ m-6 \end{array} \end{array} .$$

The center stage consists of 16 hyperbars, each of which has $M/64$ thick-wire inputs and outputs. Similarly, each message remains on the same hyperbar for the entire center switching stage. Since there are 16 hyperbars in this stage, the 4 most significant bits of the address for each message remain fixed. Thus, we characterize the center stage by a characteristic matrix C of the form

$$C = \begin{array}{cc} \begin{array}{cc} m-4 & 4 \end{array} & \\ \left[\begin{array}{c|c} \text{nonsingular} & \text{arbitrary} \\ \hline 0 & I \end{array}\right] & \begin{array}{l} m-4 \\ 4 \end{array} \end{array} .$$

We could also define a characteristic matrix form for the right switching stage, where the most significant $m-2$ bits of an address remain fixed, but we show in Appendix B that the right stage of crossbars is not necessary to perform BMMC permutations in two passes through the MasPar MP-2 network.

We characterize the wiring stages, which are a perfect shuffle of thick wires and its inverse, by the characteristic matrices

$$P = \begin{array}{cc} \begin{array}{ccc} 2 & 4 & m-6 \end{array} & \\ \left[\begin{array}{c|c|c} I & 0 & 0 \\ \hline 0 & 0 & I \\ \hline 0 & I & 0 \end{array}\right] & \begin{array}{l} 2 \\ m-6 \\ 4 \end{array} \end{array} \quad \text{and} \quad P^{-1} = \begin{array}{cc} \begin{array}{ccc} 2 & m-6 & 4 \end{array} & \\ \left[\begin{array}{c|c|c} I & 0 & 0 \\ \hline 0 & 0 & I \\ \hline 0 & I & 0 \end{array}\right] & \begin{array}{l} 2 \\ 4 \\ m-6 \end{array} \end{array} ,$$

respectively.

Appendix B shows how to decompose any nonsingular matrix into factors corresponding to at most two passes through the MP-2 global router. The overall flavor is similar to that for the EDN: we couch each switching or wiring stage as a BMMC permutation and decompose the characteristic matrix of the permutation to be routed into factors corresponding to these stages.

5 Analysis of BMMC congestion on the MasPar MP-2

Section 4 and Appendix B present an algorithm that performs any BMMC permutation in two passes through the global router of the MasPar MP-2. On the other hand, if we were to naively route an arbitrary BMMC permutation, how many passes would we need? In this section, based on some observations about the MasPar MP-2 global router, we show how to analyze the characteristic matrix to determine the probability distribution for the number of passes to naively route BMMC permutations. Finally, we show with experimental data that, in practice, the probability distribution for the number of passes to perform a BMMC permutation on the MasPar MP-2 matches the predicted probability distribution generated by linear-algebraic and combinatorial analysis.

Quantifying congestion based on the characteristic matrix

To analyze the probability distribution of arbitrary BMMC permutations in the MasPar MP-2 global router, let us first examine closely the paths of messages from their source locations to their target locations. Since $K = 4$ in the MasPar MP-2, all four messages that are destined for a particular 4×4 crossbar switch in the right stage R must have the same hyperbar in the center stage on their path. Moreover, the 4 middle bits in the target address, $y_{2..5}$, are the same for every message that exits a particular hyperbar in the center stage, i.e., all messages with the same target bits $y_{2..5}$ must pass through the same hyperbar.

Next, observe that since there is only one thick wire between each hyperbar in the left stage to a particular hyperbar in the center stage, the amount of congestion depends on the maximum number of messages that must move from some hyperbar in the left stage to the same hyperbar in the center stage. Each hyperbar in the left stage can contribute up to $K = 4$ messages to a single hyperbar in the center stage at one time. Network congestion occurs when more than K messages from the same hyperbar in the left stage must travel through the same thick wire to a hyperbar in the center stage. Since there are 64 source messages entering each hyperbar in the left stage and the most significant $m - 6$ bits are fixed on each such hyperbar, we determine the congestion on each thick wire by the mapping of the least significant 6 bits of the source address ($x_{0..5}$) to determine the middle 4 bits of the target address ($y_{2..5}$).

Using the range and preimage to determine the number of passes

Along with our above observations, we use the definition of the range and preimage to determine the number of passes for a particular BMMC permutation. For a $p \times q$ matrix A with 0-1 entries, we define the *range* of A by

$$\mathcal{R}(A) = \{y : y = Ax \text{ for some } x \in \{0, 1, \ldots, 2^q - 1\}\} ,$$

that is, $\mathcal{R}(A)$ is the set of p-vectors that can be produced by multiplying all q-vectors with 0-1 entries (interpreted as integers in $\{0, 1, \ldots, 2^q - 1\}$) by A over $GF(2)$. The following lemma from [CSW94] tells us the cardinality of the range for a given BMMC permutation.

Lemma 1 *Let A be a $p \times q$ matrix whose entries are drawn from $\{0,1\}$, let c be any p-vector whose entries are drawn from $\{0,1\}$, and let $r = \operatorname{rank} A$. Then $|\mathcal{R}(A)| = 2^r$.* ■

Thus, by our above observations and Lemma 1, the source messages entering a particular hyperbar in the left stage move to $2^{\operatorname{rank} A_{2..5,0..5}}$ different hyperbars in the center stage.

Finally, we determine the number of passes by using the definition of the preimage and another lemma from [CSW94]. For a $p \times q$ matrix A and a p-vector $y \in \mathcal{R}(A)$, we define the *preimage* of y under A by

$$\operatorname{Pre}(A, y) = \{x : Ax = y\} .$$

That is, $\operatorname{Pre}(A, y)$ is the set of q-vectors x that map to y when multiplied by A. We use the following lemma to determine how many messages move from a particular hyperbar to each hyperbar in the center stage that receives any of those messages.

Lemma 2 *Let A be a $p \times q$ matrix whose entries are drawn from $\{0,1\}$, let y be any p-vector in $\mathcal{R}(A)$, and let $r = \operatorname{rank} A$. Then $|\operatorname{Pre}(A, y)| = 2^{q-r}$.*

That is, since $2^{6-\operatorname{rank} A_{2..5,0..5}}$ messages must travel through the same thick wire to the same hyperbar in the center stage and each thick wire can carry at most $K = 4$ messages, it takes $2^{6-\operatorname{rank} A_{2..5,0..5}}/4 = 2^{4-\operatorname{rank} A_{2..5,0..5}}$ passes for all of the messages from each hyperbar in the left stage to arrive at the correct hyperbar in the center stage. Once the messages arrive at the center stage, they can move to the correct switch in the right stage without conflict.

Distribution of the number of passes on the MasPar-2

Here, we present the equation to calculate the probability for each possible rank of the submatrix $A_{2..5,0..5}$, which is the leftmost 6 columns of a $4 \times m$ matrix with full row rank:

$$\Pr\{\operatorname{rank} A_{2..5,0..5} = k\} = F_1(m,k) \cdot F_2(k) \cdot F_3(m,k) \cdot F_4(m) , \tag{1}$$

where

$$\begin{aligned} F_1(m,k) &= 2^{k(m-10+k)} , \\ F_2(k) &= \frac{(2^6-1)\cdots(2^{6-k+1}-1)}{(2^k-1)\cdots(2^1-1)} , \\ F_3(m,k) &= \frac{(2^{m-6}-1)\cdots(2^{m+k-9}-1)}{(2^{4-k}-1)\cdots(2^1-1)} , \\ F_4(m) &= \frac{(2^4-1)\cdots(2^1-1)}{(2^m-1)\cdots(2^{m-3}-1)} . \end{aligned}$$

The derivation of equation (1) requires combinatoric theory beyond the scope of this extended abstract.

Comparison to empirical results on the MasPar MP-2

In Table 1, we compare experimental data to the theoretical probabilities that an arbitrary BMMC permutation requires a certain number of passes. For each size of MasPar MP-2 (e.g., 2K processors), we generated 100,000 random BMMC permutations and counted the number of times a BMMC permutation required each number of passes. (We generated each permutation by producing $\lg^2 M + \lg M$ bits using the Unix `random()` function and testing the matrix so generated for nonsingularity.) The probabilities in each of the 'actual' and 'predicted' columns add up to 100%. The differences between the actual distributions and the expected probabilities are very slight, except for the highly unlikely cases, where we might expect to see a higher variation between the observed and expected number of occurrences.

For all sizes of the MasPar MP-2, over 99% of BMMC permutations require just 1 or 2 passes. Furthermore, a large majority of the BMMC permutations require just a single pass. Table 2

2K PEs			
passes	actual	predicted	% error
1	0.88248	0.88199	0.056
2	0.11752	0.11811	0.500

(a)

4K PEs			
passes	actual	predicted	% error
1	0.83022	0.83001	0.025
2	0.16665	0.16674	0.054
4	0.00313	0.00324	3.400

(b)

8K PEs			
passes	actual	predicted	% error
1	0.80716	0.80565	0.187
2	0.18734	0.18882	0.784
4	0.00549	0.00551	0.363
8	0.00001	0.00002	50.0

(c)

16K PEs			
passes	actual	predicted	% error
1	0.79528	0.79384	0.181
2	0.19819	0.19935	0.582
4	0.00647	0.00678	4.57
8	0.00006	0.00003	100.0
16	0.00000	0.00000002	100.0

(d)

Table 1: Comparison of the probability distribution for the number of passes to perform an arbitrary BMMC permutation on various sizes of the MasPar MP-2. We compare the probabilities produced by 100,000 random BMMC permutations performed on the MasPar MP-2 to the expected values for these probabilities produced by theoretical analysis. In this table, we measure the percentage of error as $\frac{|\text{actual} - \text{predicted}|}{\text{predicted}} \times 100\%$.

PE's	actual	predicted	% error
2K	1.11752	1.11821	0.0617
4K	1.117604	1.117645	0.0367
8K	1.20388	1.20549	0.1336
16K	1.21802	1.21990032	0.1541

Table 2: Comparison of the expected number of passes produced by experiments to the predicted number of passes produced by theoretical analysis on various sizes of the MasPar MP-2. In this table, we measure the percentage of error as $\frac{|\text{actual} - \text{predicted}|}{\text{predicted}} \times 100\%$.

compares the expected number of passes observed in practice to our theoretical analysis. In practice, we expect an arbitrary BMMC permutation to take slightly more than a pass. Thus, for most BMMC permutations, our two-pass algorithm is not necessary. For those BMMC permutations that require more than two passes, we could invoke our two-pass BMMC algorithm.

6 Conclusions

We have applied linear-algebraic techniques to show how to perform BMMC permutations in two passes on two kinds of multistage networks: the expanded delta network and the global router of the MasPar MP-2. The factoring of a BMMC characteristic matrix provides a cleaner, more straightforward notation than in previous papers with algorithms for performing BMMC and BPC permutations on multistage networks.

We further ask the following questions:

- For the class of BMMC permutations, can we show that only $O(1)$ more stages would be necessary to extend an EDN to perform all BMMC permutations in one pass? Or do BMMC permutations require $2\log_S M/K - 1$ switching stages, which is analogous to lower bounds shown for other more restricted networks? If not, how can we take advantage of the flexibility of hyperbars and thick wires to perform BMMC permutations, as well as general permutations, more efficiently?

- Can we mathematically analyze the probability distribution for the expected number of passes to perform BMMC permutations on an EDN, as we did for the MasPar MP-2?
- Is there an algorithm, efficient enough to be computed on-line, to analyze and decompose general permutations into several permutations that can be performed without collisions, thereby reducing the number of passes for bad cases? Can we analyze the expected number of passes for general permutations?
- Can we efficiently perform other useful structured classes of permutations, e.g., torus permutations, in at most two passes?

Acknowledgments

We thank Raghu Subramanian and Brian Alleyne for their help in understanding the MasPar MP-2 global router. Also, we are grateful to Isaac Scherson, whose work is supported by NSF/MasPar Grant MIP-9205737, for allowing us to use lilliput, a 4K-processor MasPar MP-2 at the University of California at Irvine. Russ Tuck of MasPar ran our pass-counting program on 8K- and 16K-processor machines at MasPar. The purchase of cascade, the 2K-processor DECmpp 12000/Sx (equivalent to an MP-2) at Dartmouth, was made possible in part by National Science Foundation Grant CDA-9222806 and by Digital Equipment Corporation.

References

[Ben65] V.E. Benes. *Mathematical Theory of Connecting Networks and Telephone Traffic.* Academic Press, New York, 1965.

[Bru94] Kristin Bruhl. BMMC permutations on a DECmpp 12000/Sx 2000. Technical Report PCS-TR94-224, Dartmouth College Department of Computer Science, July 1994. Senior thesis.

[CB95] Thomas H. Cormen and Kristin Bruhl. Don't be too clever: Routing BMMC permutations on the MasPar MP-2. In *Proceedings of the 7th Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 288–297, July 1995.

[Clo53] C. Clos. A study of non-blocking switching networks. *Bell Systems Technical Journal*, 32:406–424, March 1953.

[Cor92] Thomas H. Cormen. *Virtual Memory for Data-Parallel Computing.* PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 1992. Available as Technical Report MIT/LCS/TR-559.

[Cor93] Thomas H. Cormen. Fast permuting in disk arrays. *Journal of Parallel and Distributed Computing*, 17(1–2):41–57, January and February 1993.

[CSW94] Thomas H. Cormen, Thomas Sundquist, and Leonard F. Wisniewski. Asymptotically tight bounds for performing BMMC permutations on parallel disk systems. Technical Report PCS-TR94-223, Dartmouth College Department of Computer Science, July 1994. Preliminary version appeared in *Proceedings of the 5th Annual ACM Symposium on Parallel Algorithms and Architectures*. Revised version to appear in *SIAM Journal on Computing.*

[EHJ94] Alan Edelman, Steve Heller, and S. Lennart Johnsson. Index transformation algorithms in a linear algebra framework. *IEEE Transactions on Parallel and Distributed Systems*, 5(12):1302–1309, December 1994.

[Kau68] W.H. Kautz. Cellular interconnection arrays. *IEEE Transactions on Computers*, c-17:443–451, May 1968.

[KS83] C.K. Kruskal and M. Snir. The performance of multistage interconnection networks for multiprocessors. *IEEE Transactions on Computers*, c-32(12), December 1983.

[KS88] John Keohane and Richard E. Stearns. Routing linear permutations through the Omega network in two passes. Technical Report 88-14, Department of Computer Science, University of Albany, State University of New York, 1988. Also appeared in the *Proceedings of the 2nd Symposium on the Frontiers of Massively Parallel Computing.*

[Law75] D. H. Lawrie. Access and alignment of data in an array processor. *IEEE Transactions on Computers*, c-24:1145–1155, December 1975.

[MW95] Sean S.B. Moore and Leonard F. Wisniewski. Complexity analysis of two permutations used by fast cosine transform algorithms. Technical Report PCS-TR95-266, Dartmouth College Department of Computer Science, October 1995.

[NS81] David Nassimi and Sartaj Sahni. A self-routing Benes network and parallel permutation algorithms. *IEEE Transactions on Computers*, c-30(5):332–340, May 1981.

[OT71] D.C. Opferman and N.T. Tsao-Wu. On a class of rearrangeable switching networks. *Bell Systems Technical Journal*, 50:1579–1618, May-June 1971.

[Par80] D. Stott Parker Jr. Notes on shuffle/exchange type switching networks. *IEEE Transactions on Computers*, c-29(3):213–222, March 1980.

[Pat81] J.H. Patel. Performance of processor-memory interconnections for multiprocessors. *IEEE Transactions on Computers*, c-30(10):771–780, October 1981.

[Pea77] Marshall C. Pease. The indirect binary n-cube microprocessor array. *IEEE Transactions on Computers*, c-26(5):458–473, May 1977.

[Ram73] H.R. Ramanujam. Decomposition of permutation networks. *IEEE Transactions on Computers*, c-22(7):639–643, July 1973.

[She95] Xiaojun Shen. Optimal realization of any BPC permutation on k-extra-stage Omega networks. *IEEE Transactions on Computers*, 44(5):714–719, May 1995.

[Sie77] Howard Jay Siegel. Analysis techniques for simd machine interconnection networks and the effects of processor address masks. *IEEE Transactions on Computers*, c-26:153–161, February 1977.

[SS93] Isaac D. Scherson and Raghu Subramanian. Efficient off-line routing of permutations on restricted access expanded delta networks. In *Proceedings of the 7th International Parallel Processing Symposium*, pages 284–290, April 1993.

[Sto71] Harold S. Stone. Parallel processing with the perfect shuffle. *IEEE Transactions on Computers*, C-20(2):153–161, February 1971.

[VB81] L.G. Valiant and G.J. Brebner. Universal schemes for parallel communication. In *Proceedings of the 13th Symposium on the Theory of Computation*, pages 263–277, 1981.

[Wak68] Abraham Waksman. A permutation network. *Journal of the ACM*, 15(1):159–163, January 1968.

[YL81] Pen-chung Yew and Duncan H. Lawrie. An easily controlled network for frequently used permutations. *IEEE Transactions on Computers*, c-30(4):296–298, April 1981.

A Factoring for the EDN

Here we show that any BMMC permutation whose characteristic matrix is all-trailing nonsingular is EDN-passable. We do so by showing how to decompose any all-trailing nonsingular matrix into a product of factors corresponding to the switching and wiring stages of an EDN, in order.

To avoid confusion with earlier notation, here we denote our $m \times m$ all-trailing nonsingular matrix by F. We ignore the complement vector for now. We partition F as

$$F = \begin{array}{c} \begin{array}{ccc} \quad k \quad & \quad s \quad & m-k-s \end{array} \\ \left[\begin{array}{c|c|c} \rho & \chi & \mu \\ \hline \theta & \phi & \gamma \\ \hline \alpha & \beta & \delta \end{array}\right] \end{array} \begin{array}{l} \\ k \\ s \\ m-k-s \end{array} .$$

We can use subsets of the bits in the source or target address of a message to specify particular locations in the EDN. For the switching stages, the least significant k bits represent the *thin-wire* *offset* of a message within a particular thick wire, the middle s bits represent the *hyperbar thick-wire number* on a particular hyperbar, and the most significant $m-k-s$ bits represent the *hyperbar number*. In the wiring stage, the thin-wire offset has the same meaning and the most significant $m-k$ bits represent the *thick-wire number* among all the wires.

We define a restricted matrix form that characterizes a collision-free BMMC permutation for a switching stage in an EDN as

$$G = \begin{array}{cc} \begin{array}{cc} \scriptstyle k+s & \scriptstyle m-k-s \end{array} & \\ \left[\begin{array}{c|c} \text{nonsingular} & \text{arbitrary} \\ \hline 0 & I \end{array}\right] & \begin{array}{l} \scriptstyle k+s \\ \scriptstyle m-k-s \end{array} \end{array} .$$

Since each message remains on the same hyperbar for each switching stage, the bottom $m-k-s$ rows perform the identity mapping on the hyperbar numbers, i.e., the hyperbar number remains the same. Since the lower left $(m-k-s) \times (k+s)$ submatrix is 0, the leading $(k+s) \times (k+s)$ submatrix must be nonsingular to guarantee that we route exactly K messages from the input thick wires to each of the S output thick wires.

The following characteristic matrix is exactly the same for each stage of wires:

$$H = \begin{array}{cc} \begin{array}{ccc} \scriptstyle k & \scriptstyle m-k-s & \scriptstyle s \end{array} & \\ \left[\begin{array}{c|c|c} I & 0 & 0 \\ \hline 0 & 0 & I \\ \hline 0 & I & 0 \end{array}\right] & \begin{array}{l} \scriptstyle k \\ \scriptstyle s \\ \scriptstyle m-k-s \end{array} \end{array} .$$

Since each message remains in the same location within its thick wire, the top k rows represent the identity mapping on the thin-wire offsets. The bottom $m-k$ rows correspond to a left rotation of the thick-wire number by s bits, i.e., the S-way shuffle of the thick wires from a column of hyperbars in the EDN.

Generating rows of the characteristic matrix

With the restrictions on the characteristic matrices for each stage, we now derive the characteristic matrices for the permutation performed at each switching stage. Since we have $\log_S(M/K) =$

$(m-k)/s$ stages of wiring and switches, for $i = 0, 1, \ldots, (m-k)/s - 1$, we represent the ith switching stage and the ith wiring stage by the characteristic matrices $G^{(i)}$ and $H^{(i)}$, respectively. Thus, if each stage performs a permutation characterized by a nonsingular matrix, we characterize the permutation performed by an entire pass through the EDN as the matrix product

$$F = G^{((m-k)/s-1)} H^{((m-k)/s-1)} \cdots G^{(1)} H^{(1)} G^{(0)} H^{(0)} .$$

As we trace through the various factors (stages) from right to left, we maintain an *accumulator matrix* Z which characterizes the permutation that we have performed so far, i.e., the product of the characteristic matrices of the stages that we have traced through. We begin the permutation with the identity matrix as our accumulator matrix, since we have not permuted any data at this point. At the end of the permutation, our accumulator matrix Z will equal our characteristic matrix.

We make three observations, one about the effect on the accumulator matrix by each matrix $H^{(i)}$, one about the effect on the accumulator matrix by each matrix $G^{(i)}$, and a final one which results from the effects of the first two observations. First, each of the characteristic matrices $H^{(i)}$, for $i = 0, 1, \ldots, (m-k)/s - 1$, rotates the bottom $m-k$ rows of the accumulator matrix downward s rows. After each wiring stage, the bottom s rows of the accumulator matrix become rows k to $k+s-1$. Second, since the bottom $m-k-s$ rows of each matrix $G^{(i)}$ perform the identity mapping, the matrix $G^{(i)}$ has no effect on the bottom $m-k-s$ rows of the accumulator matrix. Third, after we rotate the middle the rows k to $k+s$ of the accumulator downward, the contents of these rows will not change during the duration of the current pass through the EDN. Thus, for the final accumulator matrix to equal the characteristic matrix F, each matrix $G^{(i)}$ must correctly derive the next s rows of the characteristic matrix F. For now, we ignore the factor $H^{(0)}$ and we derive the submatrices θ_i, σ_i, and ν_i in the matrix

$$\widehat{F} = G^{((m-k)/s-1)} H^{((m-k)/s-1)} \cdots G^{(1)} H^{(1)} G^{(0)} ,$$

which we partition as

$$\widehat{F} = \begin{array}{c} \begin{array}{ccc} k & s & m-k-s \end{array} \\ \left[\begin{array}{c|c|c} \rho & \kappa & \omega \\ \hline \theta_{(m-k)/s-1} & \sigma_{(m-k)/s-1} & \nu_{(m-k)/s-1} \\ \hline \vdots & \vdots & \vdots \\ \hline \theta_1 & \sigma_1 & \nu_1 \\ \hline \theta_0 & \sigma_0 & \nu_0 \end{array}\right] \end{array} \begin{array}{c} \\ k \\ s \\ s \\ \vdots \\ s \end{array} .$$

For example, we would use $G^{((m-k)/s-1)}$ to derive the submatrices $\theta_{(m-k)/s-1}$, $\sigma_{(m-k)/s-1}$, and $\nu_{(m-k)/s-1}$. The submatrix $\theta_{(m-k)/s-1}$ corresponds to θ in the initial matrix F, and $\left[\begin{array}{c|c} \nu_{(m-k)/s-1} & \sigma_{(m-k)/s-1} \end{array}\right]$ corresponds to the submatrix $\left[\begin{array}{c|c} \phi & \gamma \end{array}\right]$. The omitted rightmost matrix $H^{(0)}$ rotates the rightmost $m-k$ columns of $\widehat{F}$ to the left by s rows to produce our final matrix F. Since we ignore the factor $H^{(0)}$, we shall maintain the accumulator matrix Z for the product $\widehat{F}$ rather than for F.

We now show how to derive each factor $G^{(i)}$, for $i = 0, 1, \ldots, (m-k)/s - 1$. The rightmost factor is

$$G^{(0)} = \begin{array}{c} \begin{array}{ccc} k & s & m-k-s \end{array} \\ \left[\begin{array}{c|c|c} I & 0 & 0 \\ \hline \theta_0 & \sigma_0 & \nu_0 \\ \hline 0 & 0 & I \end{array}\right] \end{array} \begin{array}{c} \\ k \\ s \\ m-k-s \end{array} ,$$

where the s rows starting at row k are the bottom s rows of $\widehat{F}$. Since $G^{(0)}$ is the rightmost factor, the accumulator matrix $Z^{(0)}$ is the also same matrix.

Before performing the ith switching stage, for $i = 1, 2, \ldots, (m-k)/s - 1$, the accumulator matrix $Z^{(i)}$ corresponding to the previous i switching stages and i wiring stages is

$$Z^{(i)} = \begin{array}{cl} \begin{array}{ccccc} k & s & m-k-(i+1)s & s & (i-1)s \end{array} & \\ \left[\begin{array}{c|c|c|c|c} I & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & I & 0 \\ \hline \theta_{1..i-1} & \sigma_{1..i-1} & \tau_{1..i-1} & \eta_{1..i-1} & \zeta_{1..i-1} \\ \hline \theta_0 & \sigma_0 & \tau_0 & \eta_0 & \zeta_0 \\ \hline 0 & 0 & I & 0 & 0 \end{array}\right] & \begin{array}{l} k \\ s \\ (i-1)s \\ s \\ m-k-(i+1)s \end{array} \end{array} .$$

We further partition each submatrix ν_i into submatrices τ_i, η_i, and ζ_i, as above, but the dimensions of these submatrices will vary for each factor. In this context, we shall use range notation to indicate a number of s-row submatrices, e.g., $\theta_{1..i-1}$.

After the ith switching stage, we would like the accumulator matrix to be the matrix

$$Z'^{(i)} = \begin{array}{cl} \begin{array}{ccccc} k & s & m-k-(i+1)s & s & (i-1)s \end{array} & \\ \left[\begin{array}{c|c|c|c|c} I & 0 & 0 & 0 & 0 \\ \hline \theta_i & \sigma_i & \tau_i & \eta_i & \zeta_i \\ \hline \theta_{1..i-1} & \sigma_{1..i-1} & \tau_{1..i-1} & \eta_{1..i-1} & \zeta_{1..i-1} \\ \hline \theta_0 & \sigma_0 & \tau_0 & \eta_0 & \zeta_0 \\ \hline 0 & 0 & I & 0 & 0 \end{array}\right] & \begin{array}{l} k \\ s \\ (i-1)s \\ s \\ m-k-(i+1)s \end{array} \end{array} .$$

That is, the s rows starting at row k will equal the next s rows of the matrix $\widehat{F}$. Thus, the permutation that we perform in the ith switching stage is the permutation characterized by $G^{(i)}$ that solves the matrix product $Z'^{(i)} = G^{(i)} Z^{(i)}$. If we characterize the matrix $G^{(i)}$ as

$$G^{(i)} = \begin{array}{cl} \begin{array}{ccccc} k & s & (i-1)s & s & m-k-(i+1)s \end{array} & \\ \left[\begin{array}{c|c|c|c|c} I & 0 & 0 & 0 & 0 \\ \hline \upsilon & \epsilon & \xi & \psi & \lambda \\ \hline 0 & 0 & I & 0 & 0 \\ \hline 0 & 0 & 0 & I & 0 \\ \hline 0 & 0 & 0 & 0 & I \end{array}\right] & \begin{array}{l} k \\ s \\ (i-1)s \\ s \\ m-k-(i+1)s \end{array} \end{array} ,$$

then we need to derive the submatrices υ, ϵ, ξ, ψ and λ.

The following equations derive the s rows of the matrix $Z'^{(i)}$ starting at row k:

$$\upsilon \oplus \xi\, \theta_{1..i-1} \oplus \psi\, \theta_0 = \theta_i \ , \tag{2}$$

$$\xi\, \sigma_{1..i-1} \oplus \psi\, \sigma_0 = \sigma_i \ , \tag{3}$$

$$\xi\, \tau_{1..i-1} \oplus \psi\, \tau_0 \oplus \lambda = \tau_i \ , \tag{4}$$

$$\epsilon \oplus \xi\, \eta_{1..i-1} \oplus \psi\, \eta_0 = \eta_i \ , \tag{5}$$

$$\xi\, \zeta_{1..i-1} \oplus \psi\, \zeta_0 = \zeta_i \ . \tag{6}$$

We can combine equations (3) and (6) to form the equation

$$\left[\begin{array}{c|c} \sigma_i & \zeta_i \end{array}\right] = \left[\begin{array}{c|c} \xi & \psi \end{array}\right] \begin{array}{cl} \begin{array}{cc} s & (i-1)s \end{array} & \\ \left[\begin{array}{c|c} \sigma_{1..i-1} & \zeta_{1..i-1} \\ \hline \sigma_0 & \zeta_0 \end{array}\right] & \begin{array}{l} (i-1)s \\ s \end{array} \end{array} .$$

Our initial assumption required that each trailing submatrix of the desired characteristic matrix F be nonsingular. The above $is \times is$ matrix is the trailing $is \times is$ submatrix of the original matrix F, except that we have rotated the columns to the right by s columns. Thus, this matrix is nonsingular, allowing us to use its inverse to derive the submatrix $\left[\begin{array}{c|c} \xi & \psi \end{array}\right]$ of the matrix $G^{(i)}$. After deriving the submatrices ξ and ψ, we can substitute in equations (2), (4), and (5) to derive the submatrices υ, λ, and ϵ, respectively.

We use this method to derive each submatrix $G^{(i)}$, for $i = 0, 1, \ldots, (m-k)/s - 1$. After multiplying the accumulator matrix $Z^{(i)}$ by $G^{(i)}$ to produce $Z'^{(i)}$, we then multiply $Z'^{(i)}$ by $H^{(i+1)}$ (except for when $i = (m-k)/s - 1$). Thus multiplication rotates the bottom $m-k$ rows of $Z'^{(i)}$ downward by s rows, and so the product $H^{(i+1)}\, Z'^{(i)}$ equals $Z^{(i+1)}$. On the final switching stage, we additionally perform the appropriate permutation of the messages within each hyperbar according to the top k rows of the desired characteristic matrix F. Up to this point, all the messages reside at an address with the correct final hyperbar number.

We now extend our notation to account for a complement vector c as well as the characteristic matrix F. In addition to the accumulator matrix Z, we maintain an accumulator vector z. On each switching stage i, for $i = 0, 1, \ldots, (m-k)/s - 1$, we derive new accumulator matrix/vector pairs by multiplying both the accumulator matrix and the accumulator vector by $G^{(i)}$, as above, and by complementing the result by a complement vector $c^{(i)}$:

$$Z'^{(i)} \oplus z'^{(i)} = G^{(i)}\,(Z^{(i)} \oplus z^{(i)}) \oplus c^{(i)}\,.$$

Since $Z'^{(i)}$ equals the product $G^{(i)}\,Z^{(i)}$ as above, we set the complement vector $c^{(i)}$ by the equation $c^{(i)} = G^{(i)}\,z^{(i)} \oplus z'^{(i)}$. (Each new accumulator vector $z'^{(i)}$ has the correct next s bits of the final complement vector.) On each wiring stage, the matrix $H^{(i)}$ not only rotates the rows of the accumulator matrix down s rows, but also rotates the bits of the accumulator vector down s bits. On the last switching stage, if necessary, we complement any of the least significant k bits. Therefore, after we complete the last switching stage, the accumulator vector will equal the complement vector.

B Factoring the characteristic matrix for the MasPar MP-2

Previous work by Opferman and Tsao-Wu [OT71] and Ramanujam [Ram73] indicates we can perform BMMC permutations in two passes through the global router of the MasPar MP-2. More specifically, they show how to perform an arbitrary permutation through a network which has the following three stages in the given order: a stage of q d-permuter chips, a stage of d q-permuter chips, and a third stage identical to the first. On such a network, we must connect the jth output of the ith chip of the first stage to the ith input of the jth chip in the second stage. Similarly, we connect the ith output of the jth chip of the second stage to the jth input of the ith chip in the third stage. The composition of a left stage with a center stage and another left stage on a MasPar MP-2 with up to the full 16K processors fits this description. We shall show a much simpler and efficient algorithm for performing BMMC permutations than their algorithms for performing arbitrary permutations.

As we did for the EDN, we decompose an arbitrary BMMC permutation into two permutations, each of which can be performed in one pass through the network. For now, we ignore the complement vector c. We start with a nonsingular $m \times m$ characteristic matrix A for an arbitrary BMMC permutation, which we partition as

$$A = \begin{array}{c} \begin{array}{ccc} 2 & 4 & m-6 \end{array} \\ \left[\begin{array}{c|c|c} \beta & \lambda & \mu \\ \hline \omega & \upsilon & \rho \\ \hline \gamma & \theta & \delta \end{array}\right] \end{array} \begin{array}{l} \\ 2 \\ 4 \\ m-6 \end{array} .$$

We shall factor A into the product

$$A = R^{(2)} P^{-1} C^{(2)} P L^{(2)} R^{(1)} P^{-1} C^{(1)} P L^{(1)} ,$$

where each L, C, and R factor corresponds to a left, center, or right switching stage, respectively, and the factors P and P^{-1} correspond to wiring stages. When we ignore the complement vector, three of the switching stages will not permute at all. In particular, we set $R^{(2)} = C^{(2)} = R^{(1)} = I$, so that we factor A as

$$A = L^{(2)} P^{-1} C^{(1)} P L^{(1)} .$$

We start by creating a nonsingular trailing $(m-6) \times (m-6)$ submatrix $\widehat{\delta}$. We do so by by multiplying the characteristic matrix A by a trailer matrix T, i.e., a matrix of the form

$$T = \begin{array}{c} \begin{array}{ccc} 2 & 4 & m-6 \end{array} \\ \left[\begin{array}{c|c|c} I & 0 & * \\ \hline 0 & I & * \\ \hline 0 & 0 & I \end{array}\right] \end{array} \begin{array}{l} \\ 2 \\ 4 \\ m-6 \end{array} ,$$

where an asterisk ($*$) represents a submatrix that may contain some 1s. (See [CSW94] for details of trailer matrices.) After applying the trailer matrix, we have

$$\widehat{A} = A\,T = \begin{array}{c} \begin{array}{ccc} 2 & 4 & m-6 \end{array} \\ \left[\begin{array}{c|c|c} \beta & \lambda & \widehat{\mu} \\ \hline \omega & \upsilon & \widehat{\rho} \\ \hline \gamma & \theta & \widehat{\delta} \end{array}\right] \end{array} \begin{array}{l} \\ 2 \\ 4 \\ m-6 \end{array} ,$$

where the trailing $(m-6)\times(m-6)$ submatrix $\widehat{\delta}$ is nonsingular. Furthermore, since the matrix T is self-invertible (i.e., $T = T^{-1}$), we have $A = \widehat{A}T$. The matrix T characterizes a permutation that can be performed by the left stage of hyperbars. Therefore, we assign the matrix $L^{(1)} = T$.

For an arbitrary nonsingular matrix with a nonsingular trailing $(m-6)\times(m-6)$ submatrix $\widehat{\delta}$, we can derive a matrix product of the characteristic matrices corresponding to the perfect shuffle permutation (P), the center stage ($C^{(1)}$), and the inverse of the perfect shuffle permutation (P^{-1}) that correctly maps the source address of a message to a target address in which the most significant $m-6$ bits are identical to the most significant $m-6$ bits of the final destination address of that message. That is, we derive a matrix product for which the bottom $m-6$ rows are the bottom $m-6$ rows of the characteristic matrix $\widehat{A}$. The first 6 rows of this product characterize the identity mapping of the least significant 6 bits. We form this matrix product as the following:

$$
\begin{aligned}
Q &= P^{-1}\,C^{(1)}\,P \\
&= \begin{array}{cc} \begin{array}{ccc} 2 & m-6 & 4 \end{array} & \\ \left[\begin{array}{c|c|c} I & 0 & 0 \\ \hline 0 & 0 & I \\ \hline 0 & I & 0 \end{array}\right] & \begin{array}{c} 2 \\ 4 \\ m-6 \end{array} \end{array}
\begin{array}{cc} \begin{array}{ccc} 2 & m-6 & 4 \end{array} & \\ \left[\begin{array}{c|c|c} I & 0 & 0 \\ \hline \gamma & \widehat{\delta} & \theta \\ \hline 0 & 0 & I \end{array}\right] & \begin{array}{c} 2 \\ m-6 \\ 4 \end{array} \end{array}
\begin{array}{cc} \begin{array}{ccc} 2 & 4 & m-6 \end{array} & \\ \left[\begin{array}{c|c|c} I & 0 & 0 \\ \hline 0 & 0 & I \\ \hline 0 & I & 0 \end{array}\right] & \begin{array}{c} 2 \\ m-6 \\ 4 \end{array} \end{array} \\
&= \begin{array}{cc} \begin{array}{ccc} 2 & 4 & m-6 \end{array} & \\ \left[\begin{array}{c|c|c} I & 0 & 0 \\ \hline 0 & I & 0 \\ \hline \gamma & \theta & \widehat{\delta} \end{array}\right] & \begin{array}{c} 2 \\ 4 \\ m-6 \end{array} \end{array} \quad .
\end{aligned}
$$

The characteristic matrix Q is nonsingular because each of the factors P^{-1}, $C^{(1)}$, and P is nonsingular. The inverse of the matrix Q is

$$
Q^{-1} = \begin{array}{cc} \begin{array}{ccc} 2 & 4 & m-6 \end{array} & \\ \left[\begin{array}{c|c|c} I & 0 & 0 \\ \hline 0 & I & 0 \\ \hline \widehat{\delta}^{-1}\gamma & \widehat{\delta}^{-1}\theta & \widehat{\delta}^{-1} \end{array}\right] & \begin{array}{c} 2 \\ 4 \\ m-6 \end{array} \end{array} \quad .
$$

By multiplying $\widehat{A}$ by the inverse of the matrix Q, we derive a characteristic matrix that conforms to the restrictions for a left stage of switches. Thus, we use another pass through the left stage of switches to perform the permutation characterized by the matrix

$$\widehat{A}\,Q^{-1} = L^{(2)} = \begin{array}{c} \begin{array}{ccc} 2 & 4 & m-6 \end{array} \\ \left[\begin{array}{c|c|c} \beta \oplus \widehat{\mu}\widehat{\delta}^{-1}\gamma & \lambda \oplus \widehat{\mu}\widehat{\delta}^{-1}\theta & \widehat{\mu}\widehat{\delta}^{-1} \\ \hline \omega \oplus \widehat{\rho}\widehat{\delta}^{-1}\gamma & \upsilon \oplus \widehat{\rho}\widehat{\delta}^{-1}\theta & \widehat{\rho}\widehat{\delta}^{-1} \\ \hline 0 & 0 & I \end{array}\right] \end{array} \begin{array}{l} 2 \\ 4 \\ m-6 \end{array} .$$

If we multiply all of the factors together, we have the matrix product

$$\widehat{A} = L^{(2)}\,P^{-1}\,C^{(1)}\,P \ .$$

Our factorization for the matrix A now becomes

$$\begin{aligned} A &= \widehat{A}\,T \\ &= \widehat{A}\,L^{(1)} \\ &= L^{(2)}\,P^{-1}\,C^{(1)}\,P\,L^{(1)} \ , \end{aligned}$$

as we wished to show.

After performing the permutation characterized by the matrix A, we can apply the complement vector by complementing any of the bits of the resulting target addresses within the left stage $L^{(2)}$ and an additional center stage $C^{(2)}$. We represent the complementations performed by $L^{(2)}$ and $C^{(2)}$ as $L^{(c)}$ and $C^{(c)}$, respectively. Thus, we perform the whole BMMC permutation $\pi_{A,c}$ with characteristic matrix A and complement vector c as the composition

$$\pi_{A,c} = P^{-1} \circ C^{(c)} \circ P \circ (L^{(c)} \circ L^{(2)}) \circ P^{-1} \circ C^{(1)} \circ P \circ L^{(1)} \ .$$