ISSN 0333-3590

An Algebraic Theory for Regular Data Mappings on Parallel Computers

Hans Munthe-Kaas [1]

REPORT NO 102 MARCH 1995

Abstract. An algebraic theory for regular data mappings of multidimensional meshes is introduced. Mappings with nice algebraic properties are discussed. These include families of mesh–to–mesh permutations and reshapes of a mesh onto meshes of different shape, and are useful for expressing various regular algorithms such as e.g. dense matrix operations, fast Fourier– and wavelet–transforms, convolutions and multigrid techniques. The algebraic approach is finally used in the construction of fast optimal routing algorithms for the resulting permutations on a wide range of parallel computer networks.

1. Introduction and background. Among all data structures in parallel computing, the concept of a regular 'grid', 'mesh' or 'lattice' is by far the most important. This data structure is the basis for an almost endless list of algorithms such as e.g. dense matrix operations, fast Fourier– and wavelet–transforms, convolution and image processing algorithms, spectral and finite difference methods for PDEs, multigrid techniques and lattice–gas techniques.

Regular meshes can also be used in the study of the most common hardware devices used in parallel computing, such as e.g. mesh connected–, hypercube–, shuffle–exchange–, Ω–, butterfly– and Beneš–networks. A given data $\rightarrow$ data permutation and a set of data $\rightarrow$ device mappings induce a permutation on the device space. This introduces the important routing problem: How can we in a

[1] Department of Informatics, University of Bergen, Høyteknologisenteret, N-5020 Bergen.

given device space perform a given permutation optimally, i.e. with a minimal amount of traffic contentions? It is known via the Slepian–Duguid theorem [1, 10, 15] that many of these routing problems has an optimal solution for arbitrary permutations. The problem with this theory is that for general permutations the routings are generally too expensive to compute, unless the same permutation is to be repeated many times. This leads us to a search for restricted classes of permutations for which *fast* optimal routing algorithms exist.

In [6], a very useful framework for describing data $\to$ device mappings via so called K–tile mappings is introduced. These seem to cover most of the mappings that are in frequent use. One of the main difficulties in devising good routing algorithms for the resulting device permutations is that the literature on the subject lack good notations and a proper mathematical understanding of the mappings involved.

We have in earlier works [12, 14] shown that algebraic description of meshes (grids) is useful for the development of parallel algorithms such as fast Fourier transforms, cyclic reductions and fast elliptic solvers. In these works the additive structure (i.e. the set of shift permutations) is studied via finite Abelian groups. In this paper we add a multiplicative structure, and use the algebra of finite commutative rings to study regular meshes. Based on algebraic 'niceness' properties, we derive a set of mappings that we believe are sufficient for describing the main communication patterns of all of the above mentioned classes of algorithms. The multiplicative structure is essential in the development of fast routing algorithms. The main result of the paper is, except from the general framework introduced, a set of fast optimal routing algorithms for all the nice data $\to$ data mappings introduced, in the case where the data $\to$ device mappings are K–tile mappings.

To keep the size of the paper reasonable, we do not prove all the results in the paper. The proofs that are omitted are either likely to be found elsewhere in the literature, or can be reconstructed without substantial efforts.

2. Meshes, permutations and reshapes. Let $\mathcal{Z}_n$ denote the ring of integers $\{0, 1, 2, \ldots, n-1\}$ under addition and multiplication modulo n. We call $\mathcal{Z}_n$ a 1–dimensional mesh of size n. Let % denote the modulo function, normalized such that $0 \le z\%n < n$ for all integers z. Multidimensional meshes are defined as:

DEFINITION 1. *A d–dimensional* mesh *is a ring $\mathcal{M} = \mathcal{Z}_{n_0} \otimes \mathcal{Z}_{n_1} \otimes \mathcal{Z}_{n_2} \otimes \cdots \otimes \mathcal{Z}_{n_{d-1}}$, i.e. the set of d–dimensional column vectors $\mathbf{x} = (x_0, x_1, \ldots, x_{d-1})^T$ where x_i are integers in the range $0 \le x_i < n_i$, and where additions and multiplications are componentwise modulo n_i:*

$$\begin{aligned}
(x_0, \ldots, x_{d-1})^T + \left(x'_0, \ldots, x'_{d-1}\right)^T &= \left((x_0 + x'_0)\%n_0, \ldots, (x_{d-1} + x'_{d-1})\%n_d\right)^T \\
(x_0, \ldots, x_{d-1})^T \cdot \left(x'_0, \ldots, x'_{d-1}\right)^T &= \left((x_0 \cdot x'_0)\%n_0, \ldots, (x_{d-1} \cdot x'_{d-1})\%n_d\right)^T
\end{aligned}$$

We will use the shorthand $\mathcal{M} = \mathrm{mesh}(n_0, n_1, \ldots, n_{d-1})$ to denote the mesh. We will write $|\mathcal{M}|$ to denote the 1–dimensional mesh with the same cardinality as $\mathcal{M}$:

$$|\mathcal{M}| \equiv \mathrm{mesh}(\prod_{i=0}^{d-1} n_i) \ .$$

And if $\mathcal{M}_i = \mathrm{mesh}(n_0^i, \ldots, n_{d_i-1}^i)$ for $i = 0, 1, 2, \ldots$, then

$$\mathrm{mesh}(\mathcal{M}_0, \mathcal{M}_1, \mathcal{M}_2, \ldots) \equiv \mathrm{mesh}\left(n_0^0, \ldots, n_{d_0-1}^0, n_0^1, \ldots, n_{d_1-1}^1, n_0^2, \ldots\right) = \mathcal{M}_0 \otimes \mathcal{M}_1 \otimes \cdots$$

If the result of a computation yields a component x_i outside the standard range $\{0, \ldots, n_i - 1\}$, we say that this component has *wrapped.* It is then always reduced modulo n_i. When n_i is clear from the context we will omit writing $\%n_i$.

A (multiplicative) *inverse* of an element $\mathbf{x} \in \mathcal{M}$ is an element $\mathbf{y} \in \mathcal{M}$ such that $\mathbf{y} \cdot \mathbf{x} = (1,1,1,\ldots,1)^T$. In the 1–dimensional ring $\mathcal{M} = \mathcal{Z}_n$, an element x has an inverse, denoted $x^{-1,n}$, iff $\gcd(x,n) = 1$, i.e. x and n are relative prime. The inverse $x^{-1,n}$ is computed via the Euclidean algorithm [8]. Given x and n this algorithm computes three numbers g, c and d such that

$$c \cdot x + d \cdot n = g = \gcd(x,n)$$

thus if $g = 1$ we see that $x^{-1,n} = c\%n$. For multidimensional meshes $\mathcal{M}$, inverses are computed componentwise: $\mathbf{x}^{-1} = \left(x_0^{-1,n_0}, \ldots, x_{d-1}^{-1,n_{d-1}}\right)$, and it exists iff $\gcd(x_i, n_i) = 1$ for all i. If $\mathbf{x}^{-1}$ exist, then $\mathbf{x}$ is called a *unit* of $\mathcal{M}$.

The mesh should be understood as the address space for some data array A. *Mesh permutations* are bijective mappings of a mesh onto itself. *Mesh reshapes* are bijective mappings of a mesh onto another mesh. A permutation $P : \mathcal{M} \to \mathcal{M}$ induces a permutation of the data array A via

$$A(\mathbf{x}) \mapsto A(P(\mathbf{x})) \quad \text{for all } \mathbf{x} \in \mathcal{M} \ .$$

Thus we think of P as a function computing destination addresses from a source address. We will not explicitly mention the data array A, and only consider permutations and reshapes as mappings between meshes.

The algebraic structure of $\mathcal{M}$ induces natural permutations on $\mathcal{M}$; we let the *shift permutations* $S_{\mathbf{y}} : \mathcal{M} \to \mathcal{M}$ be defined as:

$$S_{\mathbf{y}}(\mathbf{x}) = \mathbf{x} + \mathbf{y} \text{ for all } \mathbf{x}, \mathbf{y} \in \mathcal{M} \ . \tag{1}$$

Since the arithmetic in each component i is modulo n_i, these shifts are cyclic with periode n_i in the direction i. Thus a shift of the following type: $S_{(0,0,\ldots,1,\ldots,0)^T}$ represents a nearest neighbour cyclic shift in direction i. Similarly, we can define multiplicative permutations from the multiplicative structure of $\mathcal{M}$.

2.1. Linear– and matrix reshapes. Let us for a while forget about the multiplicative structure of $\mathcal{M}$. The mesh $\langle \mathcal{M}, + \rangle$ forms a linear vectorspace[2] over the integers $\mathcal{Z}$ with vector addition given by the ring addition $+$, and multiplication by scalar $\alpha \in \mathcal{Z}$ given by

$$\alpha \cdot \mathbf{x} = \mathbf{x} + \mathbf{x} + \cdots + \mathbf{x} \ (\alpha \text{ times}) \ .$$

DEFINITION 2. Linear maps *are defined as functions* $L : \mathcal{M} \to \mathcal{M}'$ *such that* $L(\mathbf{x} + \mathbf{y}) = L(\mathbf{x}) + L(\mathbf{y})$. *A map of the form* $S_{\mathbf{y}} \circ L$ *is called an* affine linear map*.*

Let $\mathcal{M} = \text{mesh}(n_0, \ldots, n_{d-1})$ and $\mathcal{M}' = \text{mesh}(n'_0, \ldots, n'_{d'-1})$. Since $\mathcal{Z}$ is a P.I.D.[3], a linear map $L : \mathcal{M} \to \mathcal{M}'$ can be written in matrix form

$$L(\mathbf{x}) = L \cdot \mathbf{x} = \begin{pmatrix} l_{0,0} & \cdots & l_{0,d-1} \\ \vdots & \ddots & \vdots \\ l_{d'-1,0} & \cdots & l_{d'-1,d-1} \end{pmatrix} \cdot \begin{pmatrix} x_0 \\ \vdots \\ x_{d-1} \end{pmatrix},$$

where L is an integer matrix. The columns of the matrix represent vectors in $\mathcal{M}'$, and therefore we will always assume that $0 \le l_{i,j} < m_i$. We use the same letter L for both the map and its matrix. An affine linear map can be written $S_{\mathbf{y}} \circ L(\mathbf{x}) = L \cdot \mathbf{x} + \mathbf{y}$.

[2] Correctly, it should be called a $\mathcal{Z}$–module rather than a vectorspace, since $\mathcal{Z}$ is not a field.
[3] Principal Ideal Domain [5]

LEMMA 2.1. *Bijective linear– and affine linear maps exist between two meshes $\mathcal{M}$ and $\mathcal{M}'$ iff $\mathcal{M}$ and $\mathcal{M}'$ are isomorphic. (See Lemma 2.5, below).*

Since $\mathcal{Z}$ is not a field, some caution must be applied in matrix computations, compared to e.g. linear algebra over the reals. Not every integer matrix L represents a linear map:

LEMMA 2.2. *A matrix $L \in \mathcal{Z}^{d' \times d}$ represents a linear map $L : \mathcal{M} \to \mathcal{M}'$ iff*

$$(l_{i,j} \cdot n_j)\% n'_i = 0 \text{ for all } i, j \tag{2}$$

L is obviously linear for vectors $\mathbf{x}$ and $\mathbf{y}$ if $\mathbf{x}+\mathbf{y}$ does not wrap in any components. If it wraps, the condition ensures that any wrap effect is ignored.

DEFINITION 3. *A general integer matrix $M \in \mathcal{Z}^{d' \times d}$ without the linearity condition 2, is called a* matrix map*.*

A matrix map M is bijective iff its columns generates (spans) $\mathcal{M}'$ and $|\mathcal{M}| = |\mathcal{M}'|$. The inverse M^{-1} is, however, not generally a matrix map. Furthermore it is not generally true that the composition of two matrix maps is a matrix map. We will use $\circ$ to denote composition, and $\cdot$ to denote matrix product.

LEMMA 2.3. *Let $\mathcal{M} = mesh(n_0, \ldots, n_{d-1})$, $\mathcal{M}' = mesh(n'_0, \ldots, n'_{d'-1})$ and $\mathcal{M}'' = mesh(n''_0, \ldots, n''_{d''-1})$.*

1. *The inverse of a linear map L is linear. The composition of two linear maps is linear.*
2. *If $M : \mathcal{M} \to \mathcal{M}'$ and $M' : \mathcal{M}' \to \mathcal{M}''$ are matrix maps, and if the following 'no–wrap' condition holds on M:*

$$\sum_{j=0}^{d-1} m_{i,j} \cdot (n_j - 1) = n'_i - 1 \text{ for all } i, \tag{3}$$

   where $0 \le m_{i,j} < n'_i$, then the compostion is a matrix map and $M' \circ M = M' \cdot M$.
3. *If $L : \mathcal{M}' \to \mathcal{M}''$ is linear, then $L \circ M = L \cdot M$.*

An *ordering* of a mesh $\mathcal{M} = \text{mesh}(n_0, \ldots, n_{d-1})$ is a permutation σ of the components $\{0, 1, \ldots, d-1\}$. The ordering $\sigma(i) = i$ for all i is called the *default ordering*. Given an ordering σ of $\mathcal{M}$, we define the *reverse order* σ^r as $\sigma^r(i) = \sigma(d-i-1)$. The *reordered mesh*, $\mathcal{M}_\sigma$, is defined as $\mathcal{M}_\sigma = \text{mesh}(n_{\sigma(0)}, n_{\sigma(1)}, \ldots, n_{\sigma(d-1)})$, and we define the *component reordering* map $R_\sigma : \mathcal{M} \to \mathcal{M}_\sigma$ as

$$R_\sigma((x_0, x_1, \ldots, x_{d-1})^T) = ((x_{\sigma(0)}, x_{\sigma(1)}, \ldots, x_{\sigma(d-1)})^T)\ .$$

R_σ is a linear map which is represented by a permutation matrix.

2.2. Isomorphic reshapes. We now return to the ring structure of $\mathcal{M}$, with both $+$ and $\cdot$.

DEFINITION 4. *A bijection $I : \mathcal{M} \to \mathcal{M}'$ is a* ring isomorphism *if it has the properties:*

$$I(\mathbf{x}+\mathbf{y}) = I(\mathbf{x}) + I(\mathbf{y}) \tag{4}$$

$$I(\mathbf{x} \cdot \mathbf{y}) = I(\mathbf{x}) \cdot I(\mathbf{y}) \tag{5}$$

If such I exists, we say that $\mathcal{M}$ and $\mathcal{M}'$ are isomorphic.

An isomorphism is of course linear, but the converse is generally not true. The simplest example of an isomorphism is the reordering map R_σ.

The basic result for checking if two meshes are isomorphic is the *Chinese Remainder Theorem* [8][4]:

[4] After Sun Tsŭ, A.D. 350.

THEOREM 2.4. *Let $\mathcal{M} = mesh(n_0, n_1, \ldots, n_{d-1})$. The meshes $\mathcal{M}$ and $|\mathcal{M}|$ are isomorphic iff $gcd(n_i, n_j) = 1$ for all $i \neq j$. In this case we say that $\mathcal{M}$ is a* relative prime *factorization of $|\mathcal{M}|$. The isomorphism $C : |\mathcal{M}| \to \mathcal{M}$ is given in matrix form as:*

$$C = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix},$$

and the inverse as:

$$C^{-1} = \left((N/n_0)\cdot(N/n_0)^{-1,n_0}, (N/n_1)\cdot(N/n_1)^{-1,n_1} \ldots, (N/n_{d-1})\cdot(N/n_{d-1})^{-1,n_{d-1}}\right) .$$

C is henceforth called a *Chinese reshape*. We extend the definition of C to the case where $\mathcal{M}$ is a general mesh, and $\mathcal{M}'$ is obtained by relative prime factorizations of each component of $\mathcal{M}$. If all the periodes of $\mathcal{M}'$ are powers of primes, we say that $\mathcal{M}'$ is the *prime power isomorphic factorization* of $\mathcal{M}$.

LEMMA 2.5. *Two meshes $\mathcal{M}$ and $\mathcal{M}'$ are isomorphic iff their prime power isomorphic factorizations are identical, up to a reordering of the components.*

Thus any isomorphism can be constructed by composing Chinese, inverse Chinese and component reordering maps.

2.3. Hierarchical reshapes. Linear and isomorhphic reshapes are algebraically the nicest reshapes. These can only be defined between isomorphic meshes. There is, however, frequently a need for defining reshapes between non–isomorphic meshes of the same size. This need arises both in algorithms (taking meshes apart and piecing them together, see Section 3, and in data $\to$ processor mappings. Among the non–linear reshapes, the algebraically nicest are the matrix mappings which satisfy the no–wrap condition.

DEFINITION 5. *A* hierarchical reshape *is a matrix map $H : \mathcal{M} \to \mathcal{M}'$, which is a bijection and satisfy the no–wrap condition 3.*

LEMMA 2.6. *The product of two hierarchical reshapes is a hierarchical reshape.*

The reordering map R_σ is a special example of a hierarchical reshape. The 'obvious way' of mapping a multidimensional mesh into a 1–dimensional mesh of the same size is also a hierarchical map: If $\mathcal{M} = \text{mesh}(n_0, n_1, \ldots, n_{d-1})$, then the matrix map $H_{\mathcal{M}}^{|\mathcal{M}|} : \mathcal{M} \to |\mathcal{M}|$ given as

$$H_{\mathcal{M}}^{|\mathcal{M}|} = \begin{pmatrix} 1, & n_0, & n_0\cdot n_1, & \ldots, & n_0\cdot n_1 \cdots n_{d-2} \end{pmatrix} \tag{6}$$

defines a hierarchical reshape. If $\mathbf{x}' = H_{\mathcal{M}}^{|\mathcal{M}|}((x_0, \ldots, x_{d-1})^T)$, then the inverse mapping is given as:

$$x_i = \left\lfloor \frac{\mathbf{x}'}{N_i} \right\rfloor \% n_i$$

where

$$N_0 = 1 \quad \text{and } N_i = \prod_{j=0}^{i-1} n_j \text{ for all } i > 0.$$

The mappings in these two examples are the basic building blocks for building any hierarchical reshape.

DEFINITION 6. *If* $\mathcal{M} = mesh(\mathcal{M}_0, \mathcal{M}_1, \ldots, \mathcal{M}_{r-1})$ *and* $\mathcal{M}' = mesh(|\mathcal{M}_0|, \ldots, |\mathcal{M}_{r-1}|)$ *then we say that* $\mathcal{M}$ *is a* factorization *of* $\mathcal{M}'$. *In this case, we define the* default hierarchical mapping $H_{\mathcal{M}}^{\mathcal{M}'} : \mathcal{M} \to \mathcal{M}'$ *as:*

$$H_{\mathcal{M}}^{\mathcal{M}'} = \begin{pmatrix} H_{\mathcal{M}_0}^{|\mathcal{M}_0|} & 0^T & \ldots & 0^T \\ 0^T & H_{\mathcal{M}_1}^{|\mathcal{M}_1|} & \ldots & 0^T \\ \vdots & \vdots & \ddots & \vdots \\ 0^T & 0^T & \ldots & H_{\mathcal{M}_{r-1}}^{|\mathcal{M}_{r-1}|} \end{pmatrix}, \tag{7}$$

where $H_{\mathcal{M}_i}^{|\mathcal{M}_i|}$ *is defined in (6) and* 0^T *are zero row vectors of appropriate size.*

THEOREM 2.7. *If* $H : \mathcal{M} \to \mathcal{M}'$ *is a hierarchical reshape, then there exists an ordering* σ *of* $\mathcal{M}$ *such that* $\mathcal{M}_\sigma$ *is a factorization of* $\mathcal{M}'$ *and such that* $H = H_{\mathcal{M}_\sigma}^{\mathcal{M}'} \cdot R_\sigma$.

Proof: Let $\mathcal{M} = \mathrm{mesh}(n_0, n_1, \ldots, n_{d-1})$ and $\mathbf{e}_i = (0, \ldots, 1, \ldots, 0)^T$. By repeated use of the no–wrap condition and the fact that H is bijective, we conclude that we must have:

$$H^{-1}(\mathbf{e}_0) = \mathbf{e}_{\sigma(0)},\ H^{-1}(n_{\sigma(0)} \cdot \mathbf{e}_0) = \mathbf{e}_{\sigma(1)},\ H^{-1}(n_{\sigma(0)} \cdot n_{\sigma(1)} \cdot \mathbf{e}_0) = \mathbf{e}_{\sigma(2)},\ \ldots$$

When the first dimension of $\mathcal{M}'$ is exhausted, we continue with

$$H^{-1}(\mathbf{e}_1) = \mathbf{e}_{\sigma(j)},\ H^{-1}(n_{\sigma(j)} \cdot \mathbf{e}_1) = \mathbf{e}_{\sigma(j+1)},\ H^{-1}(n_{\sigma(j)} \cdot n_{\sigma(j+1)} \cdot \mathbf{e}_1) = \mathbf{e}_{\sigma(j+2)},\ \ldots$$

In this fashion we can find σ.

□

We will use the shorthand notation $H_{\mathcal{M},\sigma}^{\mathcal{M}'} \equiv H_{\mathcal{M}_\sigma}^{\mathcal{M}'} \cdot R_\sigma$. When $\mathcal{M}$ and $\mathcal{M}'$ are clear from the context, we will sometimes write H_σ instead of $H_{\mathcal{M},\sigma}^{\mathcal{M}'}$. The theorem shows that any hierarchical reshape (and thus any product of hierarchical reshapes) is characterized by the domain $\mathcal{M}$, the codomain $\mathcal{M}'$ and the ordering σ of $\mathcal{M}$.

It also shows that compositions of forwards hierarchical maps are not sufficient to construct a reshape between two arbitrary meshes with the same cardinality. This can, however, always be done if we introduce inverse hierarchical reshapes. The inverse hierarchical reshapes complicate the algebra, since the inverse of a hierarchical reshape is not a matrix map.

We will use the letters V and Λ to denote elementary compositions. The shape of these letters illustrate the order of forwards and inverse mappings. Suppose we want to reshape a mesh $\mathcal{M}'$ to a mesh $\mathcal{M}$ with the same number of points, but where neither is a factorization of the other. We may always find a mesh $\mathcal{K}$ and an ordering σ of $\mathcal{K}$ such that $\mathcal{K}$ is a factorization of $\mathcal{M}$ and $\mathcal{K}_\sigma$ is a factorization of $\mathcal{M}'$. The reshape $V_{\mathcal{M}',\mathcal{K},\sigma}^{\mathcal{M}} : \mathcal{M}' \to \mathcal{M}$ defined as:

$$V_{\mathcal{M}',\mathcal{K},\sigma}^{\mathcal{M}} = H_{\mathcal{K}}^{\mathcal{M}} \circ \left(H_{\mathcal{K},\sigma}^{\mathcal{M}'}\right)^{-1} \tag{8}$$

is called a *K-tile mapping* and the space $\mathcal{K}$ is called a *K–tile space* [6].

The letter Λ will always be used for permutations of a mesh $\mathcal{M}$ onto itself. Let σ_1, σ_2 be two orderings of $\mathcal{M}$. We define:

$$\Lambda_{\sigma_1,\sigma_2} = \left(H_{\mathcal{M},\sigma_2}^{|\mathcal{M}|}\right)^{-1} \circ H_{\mathcal{M},\sigma_1}^{|\mathcal{M}|}\ . \tag{9}$$

3. Mapping examples. In this section we will show some examples of the use of the mappings introduced above.

3.1. Linear permutations. Shift permutations are ubiquitous in applications such as e.g. convolution algorithms and finite difference algorithms. These should not need any examples. Linear permutations in general has many applications. Let $\mathcal{M} = \text{mesh}(n, n)$ be a square matrix. A transpose is given as

$$A = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} ,$$

and a counterclockwise 90 degree turn is given as the affine linear permutation

$$\begin{pmatrix} 0 & n-1 \\ 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} x_0 \\ x_1 \end{pmatrix} + \begin{pmatrix} n-1 \\ 0 \end{pmatrix} .$$

These operations can also be defined for non–square matrices, but in this case the domain and the codomain are different spaces, and the mappings are reshapes rather than permutations. In Cannon's systolic matrix multiplication algorithm, the first part of the algorithm is the 'pre–skewing' of the input matrices. This is given as the following linear permutations acting on the left and right matrices in the product:

$$A_l = \begin{pmatrix} 1 & 0 \\ n-1 & 1 \end{pmatrix} , \quad A_r = \begin{pmatrix} 1 & n-1 \\ 0 & 1 \end{pmatrix} .$$

After the preskewing, all communications of the algorithm are simple shifts.

If $\mathcal{M} = \text{mesh}(2, 2, \ldots, 2)$, i.e. $\mathcal{M}$ is a hypercube, some important linear permutations are:

$$\text{Perfect shuffle} \quad : \quad \begin{pmatrix} & & & & 1 \\ 1 & & & & \\ & 1 & & & \\ & & \ddots & & \\ & & & 1 & \end{pmatrix}$$

$$\text{Bit–reversal} \quad : \quad \begin{pmatrix} & & & & 1 \\ & & & 1 & \\ & & ⋰ & & \\ & 1 & & & \\ 1 & & & & \end{pmatrix}$$

$$\text{Binary reflected Gray–code mapping} \quad : \quad \begin{pmatrix} 1 & 1 & & & \\ & 1 & 1 & & \\ & & \ddots & \ddots & \\ & & & 1 & 1 \\ & & & & 1 \end{pmatrix}$$

3.2. Mesh splittings and mergings. Many algorithms such as e.g. multigrid techniques, fast Fourier– and wavelet transforms and cyclic reductions, are based on recursively taking meshes apart in coarser meshes, and piecing a set of coarse meshes into a fine mesh. We will briefely discuss the proper algebraic description of this process and see how this relates to the algebraically nice reshapes discussed in the previous section.

A subset $\mathcal{H}' \subset \mathcal{M}$ is called

1. A *subgroup* if $\mathbf{h}_i, \mathbf{h}_j \in \mathcal{H}' \Rightarrow \mathbf{h}_i + \mathbf{h}_j \in \mathcal{H}'$.
2. A *subring* if in addition $\mathbf{h}_i, \mathbf{h}_j \in \mathcal{H}' \Rightarrow \mathbf{h}_i \cdot \mathbf{h}_j \in \mathcal{H}'$.
3. An *ideal* if in addition $\mathbf{h} \in \mathcal{H}', \mathbf{x} \in \mathcal{M} \Rightarrow \mathbf{h} \cdot \mathbf{x} \in \mathcal{H}'$.

If $\mathcal{H}'$ is a subgroup of $\mathcal{M}$ we define the *cosets* of $\mathcal{H}'$ as the sets $\mathcal{K}_i = \mathcal{H}' + \mathbf{k}_i$ for $\mathbf{k}_i \in \mathcal{M}$. The element $\mathbf{k}_i$ is called a coset representative, and the cosets form a partitioning of $\mathcal{M}$. The cosets form the *quotient group* $\mathcal{M}/\mathcal{H}' = \{\mathcal{K}_i\}$, with addition given as $\mathcal{K}_i + \mathcal{K}_j = \mathcal{H}' + (\mathbf{k}_i + \mathbf{k}_j)$. If $\mathcal{H}'$ is an ideal we may give the quotient $\mathcal{M}/\mathcal{H}'$ a ring structure with multiplication $\mathcal{K}_i \cdot \mathcal{K}_j = \mathcal{H}' + (\mathbf{k}_i \cdot \mathbf{k}_j)$.

Let $\mathcal{H}, \mathcal{K}$ and $\mathcal{M}$ be three meshes. Suppose we have an injective linear map $H : \mathcal{H} \to \mathcal{H}' \subset \mathcal{M}$ and a surjective linear map $K : \mathcal{M} \to \mathcal{K}$ such that the kernel $\ker(K) = \mathcal{H}'$. Then $\mathcal{H}'$ must be a subgroup of $\mathcal{M}$ and $\mathcal{K} \simeq \mathcal{M}/\mathcal{H}'$. Splitting and merging of $\mathcal{M}$ in cosets of $\mathcal{H}'$ should be understood as mappings between the meshes $\mathcal{M}$ and $\mathcal{S} = \mathrm{mesh}(\mathcal{H}, \mathcal{K})$. The maps H and K give all the *important* algebraic information about the splitting of $\mathcal{M}$ into cosets of $\mathcal{H}'$. If it is important for our algorithm that the multiplicative structure of $\mathcal{M}$ is preserved under the splitting, we should choose $\mathcal{H}'$ as an ideal of $\mathcal{M}$ and let H and K be ring homomorphisms in addition to being just linear. (I.e. $H(\mathbf{h}\cdot\mathbf{h}') = H(\mathbf{h})\cdot H(\mathbf{h}')$ and $K(\mathbf{x}\cdot\mathbf{x}') = K(\mathbf{x})\cdot K(\mathbf{x}')$).

Definition 7. *Let $H, K, \mathcal{H}, \mathcal{K}$ and $\mathcal{M}$ be as above. A* merge reshape *is a reshape $A : mesh(\mathcal{H}, \mathcal{K}) \to \mathcal{M}$ such that:*

$$
\begin{aligned}
A(\mathbf{h}, \mathbf{0}) &= H(\mathbf{h}) \textit{ for all } \mathbf{h} \in \mathcal{H} \\
(K \circ A)(\mathbf{0}, \mathbf{k}) &= \mathbf{k} \textit{ for all } \mathbf{k} \in \mathcal{K} \\
A(\mathbf{h}, \mathbf{k}) &= A(\mathbf{h}, \mathbf{0}) + A(\mathbf{0}, \mathbf{k}) \textit{ for all } \mathbf{h} \in \mathcal{H}, \mathbf{k} \in \mathcal{K}
\end{aligned}
$$

A split reshape *is the inverse of a merge reshape.*

This definition does not determine A uniquely. The elements $A(\mathbf{0}, \mathbf{k})$ are the coset representatives. We are left with freedom to choose an arbitrary representative from each coset. It is natural to choose $A(\mathbf{0}, \mathbf{e}_i)$ for each basis vector $\mathbf{e}_i \in \mathcal{K}$, and extend by linearity to all of $\mathcal{K}$. In this case A becomes the matrix map where the first columns of A are the columns of H and the last columns are $A(\mathbf{0}, \mathbf{e}_i)$.

The algebraically nicest merging/splitting is the following:

Lemma 3.1. *If $mesh(\mathcal{H}, \mathcal{K})$ and $\mathcal{M}$ are isomorphic meshes, then the Chinese map $C : \mathcal{M} \to mesh(\mathcal{H}, \mathcal{K})$ is a split reshape.*

This kind of merging/splitting is the basis for twiddle–free mixed radix FFTs, but can only be used in special situations.

The situation where $\mathcal{H}'$ is an ideal of $\mathcal{M}$ is in practice the most important case, and this has also nice algebraic properties:

Lemma 3.2. *Let $\mathcal{M} = mesh(n_0, n_1, \ldots, n_{d-1})$ and $\mathcal{H}' \subset \mathcal{M}$ an ideal. In this case we may always choose*

$$
\begin{aligned}
\mathcal{H} &= mesh(n'_0, n'_1, \ldots, n'_{d-1}) \\
\mathcal{K} &= mesh(n''_0, n''_1, \ldots, n''_{d-1})
\end{aligned}
$$

where $n'_i \cdot n''_i = n_i$, and we can choose H and K as the ring homomorphisms

$$
\begin{aligned}
H &= diag(n''_0, n''_1, \ldots, n''_{d-1}) \\
K &= diag(1, 1, \ldots, 1)
\end{aligned}
$$

And, finally, we may choose the merge mapping $A : mesh(\mathcal{H},\mathcal{K}) \to \mathcal{M}$ as the hierarchical mapping:

$$A = \begin{pmatrix} n''_0 & & & & 1 & & & \\ & n''_1 & & & & 1 & & \\ & & \ddots & & & & \ddots & \\ & & & n''_{d-1} & & & & 1 \end{pmatrix}$$

When $\mathcal{H}'$ is just a subgroup and not an ideal, we must settle with a merge mapping A which is a more general matrix reshape.

EXAMPLE 1. *If $\mathcal{M} = mesh(8,8)$, $\mathcal{H} = mesh(4,4)$, $\mathcal{K} = mesh(2,2)$ and*

$$A = \begin{pmatrix} 2 & 0 & 1 & 0 \\ 0 & 2 & 0 & 1 \end{pmatrix}$$

then A^{-1} splits $\mathcal{M}$ in 4 subgrids by choosing every second point in each direction. Each subgrid maps into a subset $\mathcal{H}\otimes\mathbf{k} \subset mesh(\mathcal{H},\mathcal{K})$ for some $\mathbf{k} \in \mathcal{K}$.

EXAMPLE 2. *If $\mathcal{M} = mesh(8,8)$, $\mathcal{H} = mesh(2,2)$, $\mathcal{K} = mesh(4,4)$ and*

$$A = \begin{pmatrix} 4 & 0 & 1 & 0 \\ 0 & 4 & 0 & 1 \end{pmatrix}$$

then A^{-1} splits $\mathcal{M}$ in 4 tiles by cutting $\mathcal{M}$ along the centerline in each direction. Each tile maps into a subset $\mathbf{h}\otimes\mathcal{K} \subset mesh(\mathcal{H},\mathcal{K})$ for some $\mathbf{h} \in \mathcal{H}$.

These two examples are prototypes of multigrid– and domain decomposition type splittings, and are in a formal sense dual to each other.

EXAMPLE 3. *If $\mathcal{M} = mesh(8,8)$, $\mathcal{H} = mesh(4,8)$, $\mathcal{K} = mesh(2)$ and*

$$A = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix}$$

then A^{-1} splits $\mathcal{M}$ in 2 subgrids in a red–black checkerboard fashion. This is an example where $\mathcal{H}'$ is not *an ideal of $\mathcal{M}$.*

4. Mesh permutation models and routings. In the rest of the paper, we will derive fast optimal routing algorithms given a specific computer hardware model. Instead of studying physical interconnection networks (such as e.g. hypercubes, butterfly networks etc.), we will introduce a mesh permutation model, which describe the ability of the underlying hardware to perform permutations on a mesh.

DEFINITION 8. *A* mesh permutation model *is a pair $(\mathcal{M},\mathcal{B})$, where $\mathcal{M}$ is a mesh and $\mathcal{B}$ is a set of* basic permutations *$\mathcal{B} = \{B_i : \mathcal{M} \to \mathcal{M}\}$.*

$\mathcal{B}$ represents the permutations that can be performed efficiently on the network. General permutations must be decomposed into a product of permutations in $\mathcal{B}$. The advantage of this approach is that the functionality of the physical network is modelled via the operations on the mesh, which has a consise algebraic definition, thus we neglect details that are of minor importance to the functionality of the network.

EXAMPLE 4. *The MasPar computer [2] is a 2-D mesh consisting of 128×128 processors. The X–net hardware is capable of performing arbitrary shifts. It can be described by the network model $(\mathcal{M},\mathcal{B})$, where $\mathcal{M} = mesh(128,128)$ and $\mathcal{B} = \{\, S_{\mathbf{y}} \mid \mathbf{y} \in \mathcal{M} \,\}$.*

If we want to emphasize the fact that only nearest neighbour shifts in 8 directions can be executed fast, we may instead use the network model $(\mathcal{M}, \mathcal{B})$, where $\mathcal{M} = mesh(128, 128)$ and $\mathcal{B} = \{ S_{\mathbf{y}} \mid \mathbf{y} \in \{(0,0),(0,1),(1,1),(1,0),(1,-1),(0,-1),(-1,-1),(-1,0),(-1,1)\} \}$.

DEFINITION 9. *A* routing *of a permutation $P : \mathcal{M} \to \mathcal{M}$ is a factorization*

$$P = B_{i_r} \circ \cdots \circ B_{i_1} \circ B_{i_0} \ ,$$

where $B_i \in \mathcal{B}$. The number r is the length of the routing.

The choice of basic permutations $\mathcal{B}$ is of course crucial for the development of routing algorithms.

DEFINITION 10. *A permutation $P_i : \mathcal{M} \to \mathcal{M}$ is called an i–*coordinate permutation *if it is only affecting the value of coordinate i in $\mathcal{M}$. The set of all i–coordinate permutations is denoted $\mathcal{P}_i$.*

DEFINITION 11. *The* general coordinate permutation *model $GCP(\mathcal{M})$ is the mesh permutation model $(\mathcal{M}, \mathcal{B})$ where $\mathcal{M} = mesh(n_0, n_1, \ldots, n_{d-1})$ and $\mathcal{B} = \bigcup_{i=0}^{d-1} \mathcal{P}_i$.*

The model GCP($\mathcal{M}$) describes important aspects of a wide range of interconnection networks in frequent use.

EXAMPLE 5. *The basic functionality of Hypercube, Shuffle–exchange, Beneš, Butterfly and Ω–networks can all be described as $GCP(\mathcal{M})$, where $\mathcal{M} = mesh(2, 2, \ldots, 2)$. If we have more meshpoints than processors, and store m points in each processor, we may instead use the model $GCP(\mathcal{M})$, where $\mathcal{M} = mesh(2, 2, \ldots, 2, m)$*

There is an important difference between the hypercube and the other networks in this example. Whereas the hypercube is capable of communicating in any direction at any time (and even simultaneously in different directions), the other networks work efficiently only when the coordinates are changed in a particular order. E.g. in the shuffle–exchange network the coordinates should be modified in the order $0, 1, 2, \ldots$ or $d-1, d-2, \ldots$. Our routing algorithms will be maximally flexible with respect to restrictions on the order of which the network is capable of changing coordinates, and they will e.g. run equally well on a hypercube as on a shuffle–exchange network.

EXAMPLE 6. *The MasPar computer has a router hardware capable of performing general permutations of one point per processor. If we have m points per processor, we may describe the router hardware as $GCP(\mathcal{M})$, where $\mathcal{M} = mesh(128 \cdot 128, m)$. A more careful study of the router hardware reveals that some permutations are performed more efficiently than others. An adequate model is $GCP(\mathcal{M})$, where $\mathcal{M} = mesh(16, 64, 16, m)$. These numbers arise because there are 16 processors in a chip, 64 chips on a board and altogether 16 boards. For optimal performance there are restrictions on which order the coordinates should be changed. See [15] for details.*

The theoretical reason for our interest in the GCP model is the following result, which is a straightforward generalization of the Slepian–Duguid theorem [1, 10, 15]:

THEOREM 4.1. *Let $\mathcal{M} = mesh(n_0, n_1, \ldots, n_{d-1})$. Any permutation $P : \mathcal{M} \to \mathcal{M}$ can be factorized as*

$$P = \tilde{P}_{d-1} \circ \cdots \circ \tilde{P}_1 \circ P_0 \circ P_1 \circ \cdots \circ P_{d-1} \ ,$$

where $\tilde{P}_i, P_i \in \mathcal{P}_i$.

Thus in GCP($\mathcal{M}$) any permutation has a routing of length at most $2d-1$. It is not difficult to find examples of permutations P for which no shorter routings exist in this model, thus the routing is optimal for general permutations. The main problem with this theorem is that there are no known *fast* algorithms for computing an optimal routing for a general permutation P. It is therefore important to recognize classes of permutations where there exist fast routing algorithms. Our goal in the rest of the paper is to show that this is the case for the classes L and Λ in the case where data $\to$ processor mappings are K–tile mappings.

5. Passable and triangular permutations. In this section we will give some useful results about the GCP($\mathcal{M}$) model.

DEFINITION 12. *Let $P : \mathcal{M} \to \mathcal{M}$ be a permutation. The* components *of P is defined as $\{P_i\}_{i=0}^{d-1}$ such that*

$$P(\mathbf{x}) = (P_0(x_0), P_1(x_1), \ldots, P_{d-1}(x_{d-1}))^T \ .$$

We may regard P_i as a function of a single coordinate, or as a function $P_i : \mathcal{M} \to \mathcal{M}$ changing only the value of coordinate i. In this section we will study some permutations P where $P_i : \mathcal{M} \to \mathcal{M}$ are permutations, $P_i \in \mathcal{P}_i$. This is generally *not* the case.

DEFINITION 13. *The* independence set *of P_i, $ind(P_i)$, is the set of coordinates which P_i is not functionally depending on.*

LEMMA 5.1. *Let $P_i \in \mathcal{P}_i$ and $P_j \in \mathcal{P}_j$*

1. *If $i \in ind(P_j)$ and $j \in ind(P_i)$ then $P_i \circ P_j = P_j \circ P_i$.*
2. *If $i \in ind(P_j)$ then $P_i \circ P_j = P_j \circ P_i'$, where $P_i' = P_j^{-1} \circ P_i \circ P_j \in \mathcal{P}_i$ and $ind(P_i') \supseteq ind(P_i) \cap ind(P_j)$.*
3. *If $i \notin ind(P_j)$ and $j \notin ind(P_i)$, then it may be impossible to find $P_i' \in \mathcal{P}_i$ and $P_j' \in \mathcal{P}_j$ such that $P_i \circ P_j = P_j' \circ P_i'$.*

DEFINITION 14. *Let σ be an ordering of $\mathcal{M}$. P is called* σ–passable *if we can find $Q_i \in \mathcal{P}_i$ such that*

$$P = Q_{\sigma(d-1)} \circ Q_{\sigma(d-2)} \circ \cdots \circ Q_{\sigma(0)} \ .$$

A σ–passable permutation is easily routed in d steps by changing coordinates to the correct final value in the order $\sigma(0), \ldots \sigma(d-1)$.

DEFINITION 15. *A permutation P on $\mathcal{M}$ is:*

1. Lower triangular *if*

$$ind(P_i) = \{i+1, i+2, \ldots, d-1\} \text{ for all } i.$$

2. Upper triangular *if*

$$ind(P_i) = \{0, 1, \ldots, i-1\} \text{ for all } i.$$

3. Diagonal *if it is both lower and upper triangular.*
4. *σ–lower (upper) triangular if $R_\sigma \circ P \circ R_\sigma^{-1}$ is lower (upper) triangular on $\mathcal{M}_\sigma$.*

The importance of this definition is shown in the following lemmas:

LEMMA 5.2. *If $P : \mathcal{M} \to \mathcal{M}$ is triangular (either lower or upper), then P is σ–passable for* any *ordering σ.*

Proof: We prove the upper case: Since all P_i for $i > 0$ are independent on 0, we must have $P_0 \in \mathcal{P}_0$. After correcting the 0–coordinate to the correct final value, we may argue similarly for P_1, P_2, etc. Thus:

$$P = P_{d-1} \circ P_{d-2} \circ \cdots \circ P_0 \ .$$

Let $i > j$. Then $P_i \circ P_j = P_j' \circ P_i$ where $\mathrm{ind}(P_j') = \mathrm{ind}(P_i) \cap \mathrm{ind}(P_j) = \{0, 1, \ldots, j-1\} = \mathrm{ind}(P_j)$. Thus we may permute components according to σ without altering the independence sets.

□

LEMMA 5.3. *If P is σ–upper (lower) triangular, then P^{-1} is also σ–upper (lower) triangular.*

LEMMA 5.4. *Let R and Q be permutations on $\mathcal{M}$, and σ an ordering of $\mathcal{M}$.*

1. *If R is σ–upper triangular and Q is σ–passable, then $Q \circ R$ is σ–passable.*
2. *If R is σ–lower triangular and Q is σ–passable, then $R \circ Q$ is σ–passable.*

Proof: We prove the upper case when σ is the default ordering. Then $R = R_{d-1} \circ R_{d-2} \circ \cdots \circ R_0$, $Q = Q_{d-1} \circ Q_{d-2} \circ \cdots \circ Q_0$ and $\mathrm{ind}(R_i) = \{0, 1, \ldots, i-1\}$. Because of this independence, we may reorder the product as:

$$Q \circ R = Q_{d-1} \circ R'_{d-1} \circ \cdots \circ Q_1 \circ R'_1 \circ Q_0 \circ R_0 \ ,$$

where R'_i are derived from R_i as in Lemma 5.1. By merging $Q_i \circ R'_i$ to single i–coordinate permutations we complete the proof.

□

6. Fast optimal routing algorithms for GCP($\mathcal{M}$). By *fast*, we will mean that the cost of computing a routing for a permutation $P : \mathcal{M} \to \mathcal{M}$ is at most a polynomial in d, where d is the dimension of $\mathcal{M}$. By *optimal*, we mean that the length of the routing is at most $2 \cdot d - 1$. We will in this section develop fast optimal routing algorithms for affine linear permutations $A : \mathcal{M}_1 \to \mathcal{M}_1$ and Λ–reshapes $\Lambda : \mathcal{M}_1 \to \mathcal{M}_2$. We consider $\mathcal{M}_1$ and $\mathcal{M}_2$ as being data space meshes used in the development of a particular algorithm. In addition we have a device space mesh $\mathcal{M}$ such that $|\mathcal{M}| = |\mathcal{M}_1| = |\mathcal{M}_2|$. The device space $\mathcal{M}$ is modelling a part of the address space of the hardware where $\mathcal{M}_i$ are stored. As is done in [6], we assume that data is mapped to the hardware via K–tile mappings $V_i : \mathcal{M}_i \to \mathcal{M}$. Thus the permutation $A : \mathcal{M}_1 \to \mathcal{M}_1$ induce a permutation $V_1 \circ A \circ V_1^{-1}$ on the device space $\mathcal{M}$, and the reshape $\Lambda : \mathcal{M}_2 \to \mathcal{M}_3$ induce the permutation $V_3 \circ \Lambda \circ V_2^{-1}$ on $\mathcal{M}$.

Our goal is to prove the following theorem:

THEOREM 6.1. *Let $\mathcal{M}$, $\mathcal{M}_1$ and $\mathcal{M}_2$ be arbitrary meshes such that $|\mathcal{M}| = |\mathcal{M}_1| = |\mathcal{M}_2|$. Let $V_i : \mathcal{M}_i \to \mathcal{M}$ be arbitrary K–tile mappings. Let η_1 and η_2 be two arbitrary orderings on $\mathcal{M}$.*

If $A : \mathcal{M}_1 \to \mathcal{M}_1$ is an affine linear permutation then there is a fast algorithm computing two permutations P_1 and P_2 on $\mathcal{M}$ such that

1. *P_1 is η_1–passable and P_2 is η_2–passable in $GCP(\mathcal{M})$.*
2. *$V_1 \circ A \circ V_1^{-1} = P_2 \circ P_1$.*

If $\Lambda : \mathcal{M}_1 \to \mathcal{M}_2$ is a Λ–reshape, then there is a fast algorithm computing two permutations P_1 and P_2 on $\mathcal{M}$ such that

1. *P_1 is η_1–passable and P_2 is η_2–passable in $GCP(\mathcal{M})$.*
2. *$V_2 \circ \Lambda \circ V_1^{-1} = P_2 \circ P_1$.*

If we choose the first component of η_2 equal to the last component of η_1, the length of the routings become $2d - 1$, which is optimal.

The routing algorithms and their correctness proofs are rather complicated. We will proceed through several sections.

6.1. Linear permutations. In this section we will develop routing algorithms for linear maps in the case where the data $\to$ device maps are the identity maps. In the paper [15] we presented a routing algorithm called Binary Gaussian Elimination. This algorithm works for the case $\mathcal{M} = \mathrm{mesh}(2, 2, \ldots, 2)$, and resembles a Gaussian Elimination. The basis for this algorithm is the fact that $\mathcal{Z}_2$ is a field (i.e. all non–zeros are units), thus $\langle \mathcal{M}, + \rangle$ is a proper vectorspace. Hence all familiar vector space techniques are available. The same is also true for the case $\mathcal{M} = \mathrm{mesh}(p, p, \ldots, p)$ for a prime p.

We will prove that a modified version of the algorithm works even when $\mathcal{M} = \text{mesh}(p_0^{k_0}, p_1^{k_1}, \ldots, p_{d-1}^{k_{d-1}})$, where p_i are prime. In the ring Z_{p^k}, an element x is a unit iff p does not divide x. In the classical Gaussian Elimination, pivoting is necessary to ensure that the pivot is non–zero. In the more general case, we must pivot to guarantee that the pivot is a unit. The following algorithm is a transposed version of Gauss–Jordan elimination, where pivoting is done by adding a row to the pivot row rather than swapping two rows. These modifications are done because we want all elementary Gauss matrices and pivoting operations to be linear i–coordinate permutations.

Algorithm 1. Routing algorithm for linear permutations, prime power case:

```
function [R, Q] = lin_pp_rout(M, A)
input:
   M = mesh(n_0, n_1, ..., n_{d-1}) where n_i = p_i^{k_i} and p_i are prime.
   A :  Linear permutation on M
output:
   R and Q  :  linear permutations on M such that
      1) R = R_0·R_1·...·R_{d-1} is upper triangular.
      2) Q = Q_{d-1}·...·Q_1·Q_0 is passable in the default order.
      3) A = R·Q
begin
   R = I; Q = I; for i = [0 : d-1]
      /* Pivoting */
      if p_i|A(i,i), /* pivoting is necessary */
         find j such that (j > i)&(n_j = n_i)&¬(p_i|A(j,i))
         R_i = I + e_i·e_j^T
         A(i,0 : d-1) = A(i,0 : d-1) + A(j,0 : d-1) /* A = R_i^{-1}·A /*
      else
         R_i = I
      endif
      /* Elimination of row i */
      r = A(i,i)^{-1,n_i}
      α = A(i,0 : d-1)
      Q_i = I + e_i·(r * α + (r - 2) * e_i^T)
      /* A = A·Q_i^{-1} */
      for j = [0 : i-1, i+1 : d-1] /* All columns except pivot col.  /*
         A(0 : d-1, j) = A(0 : d-1, j) - p·A(i,j)·A(0 : d-1, i)
      endfor
      A(0 : d-1, i) = A(0 : d-1, i)·r
      /* Update R and Q */
      R = R·R_i
      Q = Q_i·Q
   endfor
end
```

Correctness proof Algorithm 1 Property 4) holds because the algorithm produce a sequence of R_i and Q_i such that

$$R_{d-1}^{-1} \cdots R_0^{-1} \cdot A \cdot Q_0^{-1} \cdots Q_{d-1}^{-1} = I$$

thus $A = R_0 \cdots R_{d-1} \cdot Q_{d-1} \cdots Q_0$. Properties 1), 2) and 3) are straightforward to check.

The only thing that could go wrong is our pivoting strategy. If the pivot search (find j such that $\ldots$) succeeds, then the pivot $A(i,i)$ is modified to $A(i,i)+A(j,i)$. If p_i divides $A(i,i)$ and not $A(j,i)$, it cannot divide the sum. Hence the new pivot is a unit. It remains to show that the pivot search is always successful. Let $\mathcal{M} = \mathcal{H} \otimes \mathcal{K}$ where $\mathcal{H} = \mathrm{mesh}(p^k, p^k, \ldots, p^k)$ and $\mathcal{K} = \mathrm{mesh}(n_0, n_i, \ldots)$, where $n_i \neq p^k$ for all i. Let A be blocked accordingly:

$$A = \begin{pmatrix} A_{0,0} & A_{0,1} \\ A_{1,0} & A_{1,1} \end{pmatrix}$$

Let $\mathcal{H}' \otimes \mathbf{0} = A(\mathcal{H} \otimes \mathbf{0})$ and $\mathbf{0} \otimes \mathcal{K}' = A(\mathbf{0} \otimes \mathcal{K})$. Since a linear permutation maps a mesh to an isomorphic mesh we must have $\mathcal{H} \otimes \mathcal{K} = \mathcal{H}' \otimes \mathcal{K}'$. In the prime–power case there is only one way to pick out the subset $\mathcal{H} \otimes \mathbf{0}$ from $\mathcal{M}$ such that $\mathcal{M} = \mathcal{H} \otimes \mathcal{K}$. Therefore $\mathcal{H} \otimes \mathbf{0} = \mathcal{H}' \otimes \mathbf{0}$, and $A_{0,0}$ must be non–singular. Thus if the pivot $A(i,i)$ belongs to $\mathcal{Z}_{p^k}$, then we can always find a unit among $A(j,i)$ when j only runs over the components belonging to $\mathcal{Z}_{p^k}$.

□

ALGORITHM 2. Routing algorithm for linear permutations, general case

```
function [R, Q] = lin_rout(M, A, η)
input:
   M = mesh(n_0, n_1, ..., n_{d-1}) arbitrary mesh
   A :  Linear permutation on M
   η :  ordering on M
output:
   R, Q :  Linear permutations on M such that:
      1) R is σ--passable for any ordering σ
      2) Q is η--passable
      3) A = R·Q
begin
   let M' be the prime power isomorphic factorization of M_η
   let C : M_η → M' be the Chinese map
   let R_η : M → M_η be the component reordering map
   let A' : M' → M' be defined as:
      A' = C·R_η·A·R_η^{-1}·C^{-1}
   [R', Q'] = lin_pp_rout(M', A')
   R = R_η^{-1}·C^{-1}·R'·C·R_η
   Q = R_η^{-1}·C^{-1}·Q'·C·R_η
end
```

NOTE 1. *Since a shift permutation is diagonal, the routing is trivially extended to affine linear maps.*

Correctness proof Algorithm 2 If $C : \mathcal{M} \rightarrow \mathcal{M}'$ is a Chinese map and $P'_{i'} \in \mathcal{P}_{i'}(\mathcal{M}')$ then $C^{-1} \cdot P'_{i'} \cdot C \in \mathcal{P}_i(\mathcal{M})$ for some i. The reordering R_η is chosen so that $C^{-1} \cdot Q' \cdot C$ becomes passable in the default order in $\mathcal{M}_\eta$, thus Q is η passable in $\mathcal{M}$. Similarly, $C^{-1} \cdot R' \cdot C$ is upper triangular in $\mathcal{M}_\eta$, and is therefore passable for any order in $\mathcal{M}$.

□

6.2. Λ–Permutations. In this section we will derive a routing algorithm for Λ–permutations in the case where the device space has only prime dimensions and the data → device mappings are inverse hierarchical maps.

DEFINITION 16. *Let* $\mathcal{M} = mesh(p_0, p_1, \ldots, p_d)$ *where* p_i *are prime and let* σ *be an ordering of* $\mathcal{M}$*. Let* $\mathcal{M}' = mesh(n_0, n_1, \ldots, n_{d'})$ *be a general mesh such that* $|\mathcal{M}'| = |\mathcal{M}|$*. By the* consistent ordering *of* σ *w.r.t.* $\mathcal{M}'$ *we mean the ordering* σ^c *of* $\mathcal{M}$ *such that* $\mathcal{M}_{\sigma^c}$ *is a factorization of* $\mathcal{M}'$*, and the components of* $\mathcal{M}_{\sigma^c}$ *are selected from left to right in* $\mathcal{M}_\sigma$ *by always choosing the leftmost component that fits into* $\mathcal{M}'$*. I.e. the first component in* $\mathcal{M}_{\sigma^c}$ *is the leftmost component of* $\mathcal{M}_\sigma$ *which is a prime factor in* n_0*, the second component is the leftmost of the remaining components of* $\mathcal{M}_\sigma$ *which is among the remaining prime factors of* n_0*, etc.*

We may think of σ^c as the ordering of $\mathcal{M}$ which is as close as possible to σ, but such that $\mathcal{M}_{\sigma^c}$ is a factorization of $\mathcal{M}'$.

EXAMPLE 7. *If* $\mathcal{M}' = mesh(15, 6)$*,* $\mathcal{M} = mesh(2, 3, 3, 5)$ *and* $\sigma = (3, 0, 2, 1)$*. Then* $\mathcal{M}_\sigma = mesh(5, 2, 3, 3)$*,* $\sigma^c = (3, 2, 0, 1)$ *and* $\mathcal{M}_{\sigma^c} = mesh(5, 3, 2, 3)$*.*

ALGORITHM 3. Routing of Λ–permutations, prime case

```
function [R,Q] = Λ_p_rout(M, σ1, σ2, η)
input:
   M = mesh(p0, p1, ..., p_{d-1}) where p_i are prime.
   σ1, σ2 :  orderings on M defining the map Λ_{σ1,σ2} : M → M
   η :  ordering on M prescribing the routing order of Q.
output:
   R, Q :  Permutations on M such that
   1) R and Q are matrix maps
   2) Λ_{σ1,σ2} = R^{-1}∘Q
   3) R and R^{-1} are σ'--passable for any ordering σ'
   4) Q is η--passable
begin
   let M' be the prime power isomorphic factorization of |M|
   let C : |M| → M' be the Chinese map
   let σ1^c be the consistent ordering of σ1 w.r.t. M'
   let σ2^c be the consistent ordering of σ2 w.r.t. M'
   L = (H^{M'}_{M,σ2^c})^{-1} ∘ H^{M'}_{M,σ1^c}.  /* L is linear on M */
   [R', Q'] = lin_rout(M, L, η)
   R = (R')^{-1} ∘ (H^{M'}_{M,σ2^c})^{-1} ∘ C ∘ H^{M'}_{M,σ2}
   Q = Q' ∘ (H^{M'}_{M,σ1^c})^{-1} ∘ C ∘ H^{M'}_{M,σ1}
end
```

Correctness proof Algorithm 3: We proceed through several lemmas

LEMMA 6.2. $R \circ Q = \Lambda_{\sigma_1, \sigma_2}$.

Proof: $R \circ Q = H_{\sigma_2}^{-1} \circ C^{-1} \circ H_{\sigma_2^c} \circ R' \circ Q' \circ H_{\sigma_1^c}^{-1} \circ C \circ H_{\sigma_1} = H_{\sigma_2}^{-1} \circ H_{\sigma_1}$

□

LEMMA 6.3. $L = H_{\sigma_2^c}^{-1} \circ H_{\sigma_1^c}$ *is a component reordering map* $R_{\tilde{\sigma}}$ *for some ordering* $\tilde{\sigma}$ *permuting components with the same prime periode p.*

Proof: Since $\mathcal{M}$ is a prime mesh and $\mathcal{M}'$ is a prime power mesh, $H_{\sigma_1^c}$ is mapping components with the same prime periode p into the same component of $\mathcal{M}'$. The inverse of $H_{\sigma_2^c}$ is taking them out in a different order, thus L is a component reordering permutation.

□

The crux of the routing algorithm is the structure of matrix maps T given as:

$$T = H_{\sigma^c}^{-1} \circ C \circ H_\sigma .$$

The structure of T is most easily seen through an example which illustrate the general case:

EXAMPLE 8. *Let $\mathcal{M} = mesh(a,b,c,a,a,b,c)$ where a,b,c are distinct primes. Let $\mathcal{M}' = mesh(a^3,b^2,c^2)$. Let σ be the default order. Then the consistent ordering of σ w.r.t. $\mathcal{M}'$ is $\sigma^c = (0,3,4,1,5,2,6)$. The composed mapping T maps an element $\mathbf{x} \in \mathcal{M}$ as:*

$$\begin{aligned}
\mathbf{x} &\mapsto \\
\mathbf{x}' &= H_\sigma \cdot \mathbf{x} = x_0 + ax_1 + abx_2 + abcx_3 + a^2bcx_4 + a^3bcx_5 + a^3b^2cx_6 \mapsto \\
\mathbf{x}'' &= C \cdot \mathbf{x}' = (\mathbf{x}'\%a^3, \mathbf{x}'\%b^2, \mathbf{x}'\%c^2)^T \mapsto \\
\mathbf{x}''' &= H_{\sigma^c}^{-1} \cdot \mathbf{x}'' = \left(\left\lfloor \frac{x_0''}{a^0} \right\rfloor \%a, \left\lfloor \frac{x_1''}{b^0} \right\rfloor \%b, \left\lfloor \frac{x_2''}{c^0} \right\rfloor \%c, \left\lfloor \frac{x_0''}{a^1} \right\rfloor \%a, \left\lfloor \frac{x_0''}{a^2} \right\rfloor \%a, \left\lfloor \frac{x_1''}{b^2} \right\rfloor \%b, \left\lfloor \frac{x_2''}{c^1} \right\rfloor \%c \right)^T \\
&= \left(\left\lfloor \frac{\mathbf{x}'}{a^0} \right\rfloor \%a, \left\lfloor \frac{\mathbf{x}'}{b^0} \right\rfloor \%b, \left\lfloor \frac{\mathbf{x}'}{c^0} \right\rfloor \%c, \left\lfloor \frac{\mathbf{x}'}{a^1} \right\rfloor \%a, \left\lfloor \frac{\mathbf{x}'}{a^2} \right\rfloor \%a, \left\lfloor \frac{\mathbf{x}'}{b^2} \right\rfloor \%b, \left\lfloor \frac{\mathbf{x}'}{c^1} \right\rfloor \%c \right)^T \\
&= \left(\frac{\mathbf{x}'\%a^1 - \mathbf{x}'\%a^0}{a^0}, \frac{\mathbf{x}'\%b^1 - \mathbf{x}'\%b^0}{b^0}, \frac{\mathbf{x}'\%c^1 - \mathbf{x}'\%c^0}{c^0}, \frac{\mathbf{x}'\%a^2 - \mathbf{x}'\%a^1}{a^1}, \dots \right. \\
&\quad \left. \dots \frac{\mathbf{x}'\%a^3 - \mathbf{x}'\%a^2}{a^2}, \frac{\mathbf{x}'\%b^2 - \mathbf{x}'\%b^1}{b^1}, \frac{\mathbf{x}'\%c^2 - \mathbf{x}'\%c^1}{c^1} \right)^T
\end{aligned}$$

Hence from the last line we see that T is given as the following matrix map:

$$T = \begin{pmatrix}
1 & & & & & & \\
1 & a & & & & & \\
1 & a & b & & & & \\
 & 1 & b & bc & & & \\
 & & & & bc & & \\
 & & a & ac & a^2c & a^3c & \\
 & & & ab & a^2b & a^3b & a^3b^2
\end{pmatrix}$$

Generally the nonzeros in a line belonging to a prime p starts in the position after the last component belonging to the same prime. So in this example we have that e.g. that component 0 and 3 belongs to a, hence the nonzeros in line 3 are in positions 1, 2 and 3.

Similarly we can show that the following holds in general:

LEMMA 6.4. *Let σ be the default order on $\mathcal{M} = mesh(p_0, \dots, p_{d-1})$, let $\mathcal{M}'$ be the prime power isomorphic factorization of $|\mathcal{M}|$, let $C : |\mathcal{M}| \to \mathcal{M}'$ be the Chinese map, let σ^c be the consistent ordering of σ w.r.t. $\mathcal{M}'$ and let $T = \left(H_{\mathcal{M},\sigma^c}^{\mathcal{M}'}\right)^{-1} \circ C \circ H_{\mathcal{M},\sigma}^{|\mathcal{M}|}$. Then*

1. *T is a matrix map.*
2. *T is lower triangular.*
3. *If $\{T_i\}_{i=0}^{d-1}$ are the components of T, and if $p_i = p_j$ then $j \in ind(T_i)$ and $i \in ind(T_j)$.*

LEMMA 6.5. *Let T be as above, and reorder the components according to an ordering η:*

$$T = T_{d-1} \circ \cdots \circ T_1 \circ T_0 = T'_{\eta(d-1)} \circ \cdots \circ T'_{\eta(1)} \circ T'_{\eta(0)} \ .$$

Then if both $\eta(i)$ and $\eta(j)$ belong to the same prime p, and if $i > j$ then $\eta(j) \in ind(T'_{\eta(i)})$. We may, however, have that $T'_{\eta(j)}$ depends on $\eta(i)$.

Proof: In the default order $T_{\eta(i)}$ and $T_{\eta(j)}$ are mutually independent. Since T is lower triangular, no new dependencies can be introduced when a component moves to the left, hence if $i > j$ we must have $\eta(j) \in \mathrm{ind}(T'_{\eta(i)})$. When other components move through $T'_{\eta(j)}$, then this component may aquire a dependency on $\eta(i)$.

□

The lemma says that if we regard all the components belonging to the same prime p, then these have an *upper triangular* dependency after arbitrary rearrangements.

LEMMA 6.6. *If Q' is the η–passable linear permutation in Algorithm 3 then $Q' \circ T = Q' \cdot T$ is an η–passable matrix permutation.*

Proof: $Q \cdot T$ is a matrix map because of Lemma 2.3. Since L in Algorithm 3 is only permuting components belonging to the same prime p, we must have that $Q_{i,j} = 0$ if i and j belong to different primes. The lemma now follows by applying Lemma 5.4 to each block of components belonging to the same prime p.

□

The correctness proof is completed by applying the same line of arguments to R.

□

6.3. Lowering linear permutations. In this section we will derive routings for mappings of the type $H^{-1} \circ A \circ H$, where H is hierarchical and A is affine linear.

First we consider the prime–power case: Let $\mathcal{M}_{pp} = mesh(p_0^{k_0}, p_1^{k_1}, \ldots, p_{d-1}^{k_{d-1}})$ where p_i are prime, not necessarily distinct. Let $\mathcal{M}_p$ be a mesh with only prime periodes and σ an ordering of $\mathcal{M}_p$ such that $H_\sigma : \mathcal{M}_p \to \mathcal{M}_{pp}$ is a hierarchical mapping. Let $A : \mathcal{M}_{pp} \to \mathcal{M}_{pp}$ be a linear permutation.

DEFINITION 17. *The* linear lowering *of A w.r.t. H_σ is the linear map $L : \mathcal{M}_p \to \mathcal{M}_p$ defined the following way:*

1. *Index the coordinates of $\mathcal{M}_p$ with two indices, x_j^i, such that $i \in \{0, 1, \ldots, d-1\}$ denotes the component it maps into in $\mathcal{M}_{pp}$, and $j \in \{0, 1, \ldots, k_i - 1\}$ denotes its hierarchical position within this group.*
2. *Define L such that if $\mathbf{x}' = L(\mathbf{x})$ then*

$$x_j'^i = \sum_k A_{i,k} \cdot x_j^k \ ,$$

where the sum ranges over all k where the coordinate x_j^k exists.

LEMMA 6.7. *Let A and L be as above. Then:*

1. *L is a linear permutation.*
2. *$H_\sigma^{-1} \circ A \circ H_\sigma = T \circ L$, where T is a permutation that is lower triangular w.r.t. the ordering $\{x_0^0, x_0^1, x_0^2 \ldots, x_1^0, x_1^1, x_1^2, \ldots\}$ of $\mathcal{M}_{pp}$.*

Proof: If $p_i \neq p_j$, the linearity condition 2 gives $A_{i,j} = 0$, so there are no couplings between components over different primes. We may therefore consider the special case where $p_0 = p_1 = \ldots = p_{d-1} = p$. Let $\mathbf{x}' = \left(H_\sigma^{-1} \circ A \circ H_\sigma\right)(\mathbf{x})$. Then $x_j'^i = \left\lfloor \frac{\sum_{j,l} L_{i,j} \cdot p^l \cdot x_l^j + b_i}{p^j} \right\rfloor \% p$. All terms $A_{i,j} \cdot p^l \cdot x_l^j$ where $l > j$ vanish, and all terms where $l = j$ can be taken outside the floor function, and thus behave linearly. These are put into the matrix L. Since any matrix over $\mathcal{M}_p$ represents a linear transformation, L is a linear map. The terms where $l < j$ are the nonlinear terms, and are put into T'. Thus $(H_\sigma^{-1} \circ A \circ H_\sigma)(\mathbf{x}) = L \cdot \mathbf{x} + T'(\mathbf{x})$. If we order the components as $x_0^0, x_0^1, \ldots, x_1^0, x_1^1, \ldots, x_2^0, x_2^1, \ldots$, we see that T' has strictly block lower triangular dependencies w.r.t. this ordering, and L is strictly block diagonal. By an argument similar to the one used to guarantee the pivot strategy of Algorithm 1, we see that all the diagonal blocks of L must be non–singular, and thus L is a linear permutation. Let $T(\mathbf{x}) = \mathbf{x} + T'(L^{-1}(\mathbf{x}))$. Then $H_\sigma^{-1} \circ A \circ H_\sigma = T \circ L$. Due to the strict lower triangular block dependencies in T' and the block diagonal structure of L we see that T is lower triangular w.r.t. the given ordering.

□

The structure of the linear lowering is illustrated by the following example:

EXAMPLE 9. *Let $\mathcal{M}_{pp} = mesh(8,4,2)$ and $\mathcal{M}_p = mesh(2,2,2;2,2;2)$. Let H_σ be the default hierarchical mapping and let*

$$A = \begin{pmatrix} a_{0,0} & a_{0,1} & a_{0,2} \\ a_{1,0} & a_{1,1} & a_{1,2} \\ a_{2,0} & a_{2,1} & a_{2,2} \end{pmatrix}$$

The ordering $x_0^0, x_0^1, x_0^2, x_1^0, x_1^1, x_2^2$ corresponds to the ordering $\eta = (0,3,5,1,4,2)$ of $\mathcal{M}_p$. Under this ordering, the linear lowering of A onto $\mathcal{M}_p$ is written:

$$\begin{pmatrix} x_0' \\ x_3' \\ x_5' \\ x_1' \\ x_4' \\ x_2' \end{pmatrix} = \begin{pmatrix} a_{0,0} & a_{0,1} & a_{0,2} & & & \\ a_{1,0} & a_{1,1} & a_{1,2} & & & \\ a_{2,0} & a_{2,1} & a_{2,2} & & & \\ & & & a_{0,0} & a_{0,1} & \\ & & & a_{1,0} & a_{1,1} & \\ & & & & & a_{0,0} \end{pmatrix} \cdot \begin{pmatrix} x_0 \\ x_3 \\ x_5 \\ x_1 \\ x_4 \\ x_2 \end{pmatrix}$$

Now we can state the routing algorithm for the general case:

ALGORITHM 4. Routing algorithm for $H^{-1} \circ L \circ H$ permutations:

function $[Q] =$ lin_H_rout$(\mathcal{M}_p, \sigma, \mathcal{M}, A, \eta)$
input:
  $\mathcal{M}_p = \text{mesh}(p_0, p_1, \ldots, p_{d-1})$ where p_i are prime
  $\mathcal{M} = \text{mesh}(n_0, n_1, \ldots, n_{d'-1})$, general mesh
  σ : ordering on $\mathcal{M}_p$ s.t. $H_\sigma : \mathcal{M}_p \to \mathcal{M}$ is hierarchical
  $A : \mathcal{M} \to \mathcal{M}$: linear permutation
  η : ordering on $\mathcal{M}_p$ prescribing a routing order
output:
  Q : permutation on $\mathcal{M}_p$ such that
    1) $H_\sigma^{-1} \circ A \circ H_\sigma = R \circ Q$
    2) $Q : \mathcal{M}_p \to \mathcal{M}_p$ is a η--passable matrix permutation on $\mathcal{M}_p$
    3) $R : \mathcal{M}_p \to \mathcal{M}_p$ is σ--passable for any ordering σ on $\mathcal{M}_p$
begin
  let $\mathcal{M}_{pp}$ be the prime power isomorphic factorzation of $\mathcal{M}$
  let $C : \mathcal{M} \to \mathcal{M}_{pp}$ be the Chinese map
  let $A_{pp} = C \cdot A \cdot C^{-1}$
  let σ^c be the consistent ordering of σ w.r.t. $\mathcal{M}_{pp}$
  let $L : \mathcal{M}_p \to \mathcal{M}_p$ be the linear lowering of A_{pp} w.r.t. H_{σ^c}.
  $[R', Q'] =$ lin_rout$(\mathcal{M}_p, L, \eta)$
  $Q = Q' \circ H_{\sigma^c}^{-1} \circ C \circ H_\sigma$
end

NOTE 2. *R is* not *generally a matrix map, and does not have any nice representation except as the composition $H_\sigma^{-1} \circ A \circ H_\sigma \circ Q^{-1}$, where the terms are matrix maps.*

Correctness proof:

$H_\sigma^{-1} \circ A \circ H_\sigma = H_\sigma^{-1} \circ C^{-1} \circ A_{pp} \circ C \circ H_\sigma = H_\sigma^{-1} \circ C^{-1} \circ H_{\sigma^c} \circ T \circ R' \circ Q' \circ H_{\sigma^c}^{-1} \circ H_\sigma = T_2 \circ T \circ R' \circ Q' \circ T_1$, where T is the block lower triangular permutation arising in the linear lowering of A_{pp}. We have that $Q = Q' \cdot T_1$ is η passable by the same argument used in the proof of Algorithm 3. Furthermore, R' is block diagonal with triangular blocks under a particular ordering, and due to the block structure of T we have that $T \circ R'$ is triangular and thus passable in any order. Furthermore, $T \circ R'$ is not mixing different primes, thus $R = T_2 \circ T \circ R'$ is passable in any order, by the same argument as used in the proof of Algorithm 3.

□

6.4. Lifting routings. In this section we will derive routings for permutations of the form $H \circ P \circ H^{-1}$. Obviously, if P_i is an i–coordinate permutation on a space $\mathcal{K}$ and $H : \mathcal{K} \to \mathcal{M}$ is a hierarchical map, then $H \circ P_i \circ H^{-1}$ is a j–coordinate permutation on $\mathcal{M}$. So if we are free to route a permutation $P : \mathcal{K} \to \mathcal{K}$ in an arbitrary order, we can impose any ordering on the permutation $H \circ P \circ H^{-1}$. If a permutation is given as $(V)^{-}1 \circ A \circ V$, we start by lowering the bottom of V to a prime space $\mathcal{K}$, and then lift up again by the right leg of V. This leads to the following algorithm:

ALGORITHM 5. Routing of $V^{-1} \circ L \circ V$–permutations

function $[Q] =$ lin_V_rout$(\mathcal{M}, \mathcal{M}_1, \mathcal{K}_1, \kappa_1, A, \eta)$

input:

$\mathcal{M}$: device space. General mesh.

$\mathcal{M}_1, \mathcal{K}_1, \kappa_1$: data space $\mathcal{M}_1$ with device mapping $V^{\mathcal{M}}_{\mathcal{M}_1, \mathcal{K}_1, \kappa_1}$

$A : \mathcal{M}_1 \to \mathcal{M}_2$ matrix part of an affine linear permutation

η : ordering on $\mathcal{M}$ prescribing the routing order of Q.

output:

Q : Permutation on $\mathcal{M}$ such that

1) $V^{\mathcal{M}}_{\mathcal{M}_1, \mathcal{K}_1, \kappa_1} \circ A \circ \left(V^{\mathcal{M}}_{\mathcal{M}_1, \mathcal{K}_1, \kappa_1}\right)^{-1} = R \circ Q$

2) $Q : \mathcal{M} \to \mathcal{M}$ is an η--passable permutation

3) $R : \mathcal{M} \to \mathcal{M}$ is a σ--passable permutation for any ordering σ on $\mathcal{M}$

begin

let $\mathcal{K}$ be a prime factorization of $\mathcal{K}_1$

let μ be the default ordering of $\mathcal{K}$

let η' be the consistent ordering of μ w.r.t. $\mathcal{M}_\eta$

let σ be the ordering of $\mathcal{K}$ such that:

$H^{\mathcal{M}_1}_{\mathcal{K}_1, \kappa_1} \circ H^{\mathcal{K}_1}_{\mathcal{K}} = H^{\mathcal{M}_1}_{\mathcal{K}, \sigma}$

$[Q'] =$ lin_H_rout$(\mathcal{K}, \sigma, \mathcal{M}_1, \eta')$

$Q = H^{\mathcal{M}}_{\mathcal{K}} \circ Q' \circ \left(H^{\mathcal{M}}_{\mathcal{K}}\right)^{-1}$

end

NOTE 3. *Q is not a matrix map, and is thus not represented in a compact way other than as the composition $H^{\mathcal{M}}_{\mathcal{K}} \circ Q' \circ \left(H^{\mathcal{M}}_{\mathcal{K}}\right)^{-1}$. R does not have a compact representation other than as a composition of several (forwards and inverse) matrix maps. If A is an affine linear permutation, the shift part of A does not complicate the routing. It will just give a triangular contribution in R and can thus be neglected in the routing algorithm.*

In our final algorithm we will route the permutation $V_2 \circ \Lambda \circ V_1^{-1}$. We want to do this as above, by lowering the bottom of V_1 and V_2 to a prime space $\mathcal{K}$, and thereafter lifting along the rightmost and leftmost legs. The only problem of applying this strategy is that the two legs in general are different.

The following lemma shows that we may 'twist' the left leg of the lowered V_2 so that it coincides with the right leg of the lowered V_1^{-1}. The cost of this is the addition of a diagonal mapping D. A diagonal mapping does, however, not complicate the routing.

LEMMA 6.8. *Let $\mathcal{M}$ be an arbitrary mesh, $\mathcal{M}'$ a factorization of $\mathcal{M}$ and let $H_{\mathcal{M}'}^{\mathcal{M}}$ be the default hierarchical map. Let $\mathcal{M}_p$ be a prime mesh and σ an ordering of $\mathcal{M}_p$ such that $H_{\mathcal{M}_p,\sigma}^{\mathcal{M}}$ is hierarchical. Let σ^c be the consistent ordering of σ w.r.t. $\mathcal{M}'$. Then*

$$H_{\mathcal{M}'}^{\mathcal{M}} \circ H_{\mathcal{M}_p,\sigma^c}^{\mathcal{M}'} \circ \left(H_{\mathcal{M}_p,\sigma}^{\mathcal{M}}\right)^{-1} = D$$

where D is a diagonal permutation on $\mathcal{M}$.

Proof: D is taking components in $\mathcal{M}$ apart and putting them into the same component again in a possibly different order. On each component of $\mathcal{M}$, this mapping is independent on the other components.

□

ALGORITHM 6. Routing of $V_2 \circ \Lambda \circ V_1^{-1}$–permutations

```
function [Q] = Λ_V_rout(M, M1, K1, κ1, M2, K2, κ2, σ1, σ2, η)
input:
   M :  device space.  General mesh.
   M1, K1, κ1 :  initial data space M1 with device mapping V^M_{M1,K1,κ1}
   M2, K2, κ2 :  final data space M2 with device mapping V^M_{M2,K2,κ2}
   σ1, σ2 :  orderings on M1 and M2 defining a permutation Λ^{M2}_{M1,σ1,σ2}
   η :  ordering on M prescribing the routing order of Q.
output:
   Q :  Permutation on M such that
   1)  V^M_{M2,K2,κ2} ∘ Λ^{M2}_{M1,σ1,σ2} ∘ (V^M_{M1,K1,κ1})^{-1} = R∘Q
   2)  Q : M → M  is an η--passable permutation
   3)  R : M → M  is a σ--passable permutation for any ordering σ on M
begin
   let K be the prime factorization of K1
   let μ be the default ordering of K
   let σa be the ordering of K such that:
       H^{|M|}_{M1,σ1} ∘ H^{M1}_{K1,κ1} ∘ H^{K1}_{K,μ} = H^{|M|}_{K,σa}
   let μ^c be the consistent ordering of μ w.r.t. K2
   let σb be the ordering of K such that:
       H^{|M|}_{M2,σ2} ∘ H^{M2}_{K2,κ2} ∘ H^{K2}_{K,μ^c} = H^{|M|}_{K,σb}
   let η' be the consistent ordering of μ w.r.t. M_η
   [R', Q'] = Λ_p_rout(K, σa, σb, η')
   Q = H^M_K ∘ Q' ∘ (H^M_K)^{-1}
end
```

NOTE 4. *As in Algorithm 5, Q and R cannot be represented as simple matrix maps.*

7. A routing example. Finally, it is useful to see a concrete example of a routing.

REFERENCES

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

[2] Blank T.: *The MasPar MP–1 architecture* In *CompCon'90* San Francisco, California, 1990.

[3] Bosman O., Fletcher P., Tsui K.: *K–tiling: A structure to support regular ordering and mapping of image data*, In ARPS Workshop on Two and Three Dimensional Spatial Data: Representation and Standards, Perth, December 7–8 1992.

[4] Dahl D.: *Mapping and Compiled Communication on the Connection Machine System*, Proceedings of teh fifth distributed memory computing conference - DMCC 1990, IEEE computer society.

[5] Fraleigh : *A first course in Abstract Algebra*, ...

[6] Fletcher P. *Regular Mapping of Multidimensional Data on Parallel Processors*, Ph.D. thesis Australian National University, TR-HJ-93-05 Centre for Spatial Information Systems, Canberra, Australia (1993).

[7] Huang S-T, Tripathi S.K.: *Self–Routing Technique in Perfect-Shuffle Networks Using Control Tags*, IEEE Trans. Comput., vol. 37, pp. 251-256, no. 2 Feb. 1988.

[8] Knuth D.E.: *The Art of Computer Programming*, Vol. 1–3, Addison–Wesley, Reading Massachuchets (1973).

[9] Kuszmaul C.L.: *FFT Communications Requirement Optimizations on Massively Parallel Architectures with Local and Global Interprocessor Communications Capabilities*, MasPar report # TW009.0790, Presented at The International Society for Optical Engineering; July 8-13, 1990; San Diego, CA.

[10] Leighton F.T.: *Introduction to Parallel Algorithms and Architectures: Arrays–Trees–Hypercubes*, Morgan Kaufmann, San Mateo, CA, 1992.

[11] Lenfant J.: *Parallel Permutations of Data: A Benes Network Control Algorithm for Frequently Used Permutations*, IEEE Trans. Comput., vol. C-27,pp. 637-647, no. 7 July 1978.

[12] Munthe–Kaas H.: *Topics in Linear Algebra for Vector– and Parallel Processors*, Ph.D. Thesis, Norwegian Institute of Technology, 1989.

[13] Munthe–Kaas H.: *Generalized Shuffle Exchange Networks*, In Springer Lecture notes in comp. science vol. 634, 1992.

[14] Munthe–Kaas H.: *Super Parallel FFT's*, SIAM J. Sci. Comput., Vol. 14, No. 2, pp. 349–367, March 1993.

[15] Munthe–Kaas H.: *Practical Parallel Permutations Procedures*, to appear.

[16] Nassimi D., Sahni S.: *A Self-Routing Benes Network and Parallel Permutation Algorithms*, IEEE Trans. Comput., vol. C-30,pp. 332-340, no. 5 May 1981.

[17] Parker S.D.: *Notes on Shuffle/Exchange-Type Switching Networks*, IEEE Trans. Comput., vol. C-29,pp. 213-222, no. 3 March 1980.

[18] Pradhan D.K., Samtham M.R.: The deBruijn multiprocessor network: A versatile parallel processing and sorting network for VLSI, IEEE trans. on comput. 38 (1989).

[19] Steinberg D.: *Invariant Properties of the Shuffle-Exchange and a Simplified Cost-Effective version of the Omega Network*, IEEE Trans. Comput. vol. C-32, no. 5, May 1983.

[20] Stone H.S.: *Parallel Processing with the Perfect Shuffle*, IEEE trans. on comput. C-20, 2, (Feb 1971).