Minimization of Memory and Network Contention for Accessing Arbitrary Data Patterns in SIMD Systems

Mayez A. Al-Mouhamed* and Steven S. Seiden†

Abstract

Finding general XOR-schemes to minimize memory and network contention for accessing arrays with arbitrary sets of data *templates* is presented. A combined XOR-matrix is proposed together with a necessary and sufficient condition for conflict-free access. We present a new characterization of the baseline network. Finding an XOR-matrix for combined templates is shown to be an NP-complete problem. A heuristic is proposed for finding XOR-matrices by determining the constraints of each template-matrix and solving a set of simultaneous equations for each row. Evaluation shows significant reduction of memory and network contention compared to interleaving and to static row-column-diagonals storage.

Keywords: Memory Conflicts, multistage networks, NP-completeness, parallel memories, storage schemes.

1 Introduction

Non-uniform access to parallel memories and network contention are responsible for significant performance degradation [11, 2], especially in SIMD systems. The use of a prime number of memories [11] significantly outperforms interleaving but requires expensive [9] address translation. By restricting the type of access to some fixed data patterns, conflict-free access [6, 3, 10] to rows, columns, and diagonals of arrays were proposed on the basis of row rotations. The drawbacks are the dependence on the array size, the number of memories, and the complex address transformation.

Based on *skew storage*, *XOR-schemes* were proposed [6, 7, 14] for eliminating most of the above problems. In [7] a necessary and sufficient condition for conflict-free memory access of a given template is presented but no methods were proposed for finding the XOR-scheme for composite templates.

Because of the excessive cost of crossbar switches, multistage networks such as the Benes [4], Omega [10] and, Baseline [13] were analyzed with respect to their permissible permutations.

*Computer Engineering Department, College of Computer Science and Engineering, King Fahd University of Petroleum and Minerals, Dhahran 31261, Saudi Arabia. Email: mayez@ccse.kfupm.edu.sa

†Department of Information and Computer Science, University of California, Irvine, CA 92717.

In most cases, conflict-free access to the network is obtained for some fixed data templates. For row, column, diagonals, and square blocks, a scheme [5] based on composite linear permutations was proposed for the Omega network.

While all these approaches are useful, they either: 1) treat only the parallel memory characterization, or 2) treat memory and network aspects by considering only a reference set of static templates. Our objective is to find a storage scheme that combines the requirements of composite data templates and network topology in synthesizing global address transformation so that memory and network contentions are minimized.

This paper is organized as follows. Section 2 describes the notation used. Section 3 defines templates and XOR-schemes. Characterization of the Baseline network is presented in Section 4. The NP-completeness aspects are presented in Section 5. An algorithm for finding XOR-schemes is presented in Section 6. Section 7 presents the performance evaluation and Section 8 concludes this work.

2 Notation

We denote abstract objects such as vector spaces and templates using capital cursive letters, i.e. the vector space $\mathcal{V}$. Sets are denoted using capital italic letters, i.e. the set S. Integers and vectors are denoted using small italic letters. The binary representation of a d bit integer x is $x_{d-1} \ldots x_1 x_0$. We can identify the set of integers in the range $[0 : 2^d - 1]$ with the d dimensional vector space over Z_2, the integers modulo 2. Addition corresponds to logical **exclusive-or** and multiplication corresponds to logical **and**. Adding two vectors in Z_2^n corresponds to taking the bitwise **exclusive-or**.

Vectors of some vector space are *linearly independent* if and only if no non-zero *linear combination* of them is zero. Formally, if we have a set of vectors $\{v_0, \ldots, v_{n-1}\}$ then they are linearly independent if and only if the linear combination $a_0 v_0 \oplus \cdots \oplus a_{n-1} v_{n-1}$ is zero exactly when all a_i are zero, where each a_i is a scalar. The vector space Z_2^n is generated by its canonical basis which will be denoted by $\{v_0, \ldots, v_{n-1}\}$.

Let ϕ be a function. Then we say ϕ is *linear* if $\phi(x \oplus y) = \phi(x) \oplus \phi(y)$ and $\phi(cx) = c\phi(x)$. Suppose ϕ is a linear function $\phi : Z_2^n \mapsto Z_2^m$. The function ϕ is represented by an $m \times n$ matrix Φ over Z_2. We apply ϕ to a vector x by matrix multiplication $\phi(x) = \Phi x$. The upper left-most entry of Φ is $\Phi_{0,0}$. We denote the ith row of Φ by $\Phi_{i,*}$, and the jth column of Φ by $\Phi_{*,j}$. The columns of Φ represent the values of ϕ on the basis vectors of Z_2^n. If v_j is the jth basis vector, $\Phi_{*,j}$ is the value of $\phi(v_j)$. ϕ will be onto if and only if Φ has full rank.

Since ϕ produces a vector, we wish to consider how ϕ produces each component of that vector. We define $\phi_i(x)$ to be the ith component of the vector $\phi(x)$. Or more succinctly $\phi(x) = (\phi_0(x), \ldots, \phi_{m-1}(x))$, where $\phi_i(x) = \Phi_{i,*} x$.

Given a subset S of the basis of Z_2^n, we can create a *restriction* of ϕ to S denoted by ϕ^S. Since the basis of Z_2^n is an ordered set, an ordering of S is imposed by the ordering of the basis. The matrix representing the restricted function is denoted Φ^S. Formally, if S is a subset of the basis $v_0, \ldots, v_{n-1}$ of Z_2^n, and m is the size of S, then ϕ^S is defined by $\phi^S(x) = (\phi_{i_0}(x), \ldots, \phi_{i_{m-1}}(x))$, where v_{i_j} is the jth member of S. The matrix Φ^S consists of the columns $\Phi_{*,i_0} \ldots \Phi_{*,i_{m-1}}$, concatenated in that order.

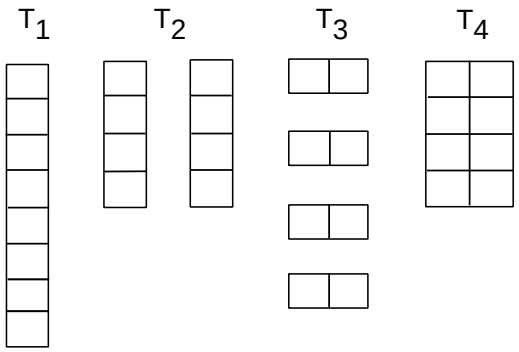

Figure 1: Example of a set of patterns

3 XOR-schemes

We wish to identify each array position (a, b) with a unique vector. The row index a can be identified with the vector $(a_0, \ldots, a_{d-1})$ and the column index b can be identified with $(b_0, \ldots, b_{d-1})$. Array position (a, b) can be uniquely identified with $(a_0, \ldots, a_{d-1}, b_0, \ldots, b_{d-1})$. We also identify each memory unit number c with $(c_0, \ldots, c_{p-1})$. We now define formally the vector spaces to which we have been alluding.

We define a vector space $\mathcal{F} = Z_2^d$ to represent horizontal indices. We define $F = \{f_0, f_1, \ldots, f_{d-1}\}$ to be the canonical basis of $\mathcal{F}$. We similarly define vector spaces $\mathcal{G}$ for column indices and $\mathcal{H}$ for memory unit numbers, with canonical bases $G = \{g_0, g_1, \ldots, g_{d-1}\}$ and $H = \{h_0, h_1, \ldots, h_{p-1}\}$, respectively.

The Cartesian product of the vector spaces $\mathcal{F}$ and $\mathcal{G}$ is a new vector space $\mathcal{V} = \mathcal{F} \times \mathcal{G}$ with basis $F \cup G$. Let $n = 2d$. We denote this combined basis as $V = \{v_0, v_1, \ldots, v_{n-1}\}$, where $v_0 = f_0$, $v_1 = f_1$, $v_d = g_0$ etc.... This vector space is isomorphic to Z_2^n. Any position (a, b) in the array is uniquely associated with a vector in $\mathcal{V}$. This vector can be expressed as a linear combination of the basis elements $a_0 v_0 \oplus \cdots \oplus a_{d-1} v_{d-1} \oplus b_0 v_d \oplus \cdots \oplus b_{d-1} v_{n-1}$. We refer to the elements of the basis V using either f's and g's, or v's, depending on which is notationally convenient.

A *template* is a pattern on array element locations. Templates are used to describe the access patterns of an algorithm. The problem is to find a scheme that allows conflict-free access to all instances of a template. XOR-schemes can handle a large class of useful templates, such as rows, columns, square, blocks, and power-of-2 strides.

A template $\mathcal{T}_i$ is defined by its basis T_i, which is a non-empty subset of V. All template bases are of size p. Figure 1 shows some templates. Let p and d both be 3. Our basis is $V = \{f_0, f_1, f_2, g_0, g_1\}$, or alternatively $V = \{v_0, v_1, \ldots, v_4\}$. Template $\mathcal{T}_1$ is defined by its basis $T_1 = \{f_0, f_1, f_2\}$. Every element in a template instance is a linear combination:

$$a_0 f_0 \oplus a_1 f_1 \oplus a_2 f_2 \oplus b_0 g_0 \oplus b_1 g_1$$

where the b's are constant, and the a's are allowed to vary. Similarly, templates $\mathcal{T}_2$, $\mathcal{T}_3$, and $\mathcal{T}_4$ have bases $T_2 = \{f_0, f_1, g_1\}$, $T_3 = \{f_1, f_2, g_0\}$, and $T_4 = \{f_0, f_1, g_0\}$, respectively.

An XOR-scheme is a linear function $\phi : \mathcal{F} \times \mathcal{G} \mapsto \mathcal{H}$, that is a $p \times n$ matrix Φ. An XOR-scheme ϕ allows conflict-free access to $\mathcal{T}_i$ if and only if ϕ maps each linear combination

of T_i to a unique element of $\mathcal{H}$ [7]. In other words, ϕ^{T_i} must be onto. For conflict-free access to all templates, ϕ^{T_i} must be onto for all T_i.

4 The Baseline Network

We consider SIMD systems in which the PE's are interconnected to the memories by using a *Baseline network* as the data-alignment system. We study how a message passes through a p-stage Baseline network from a given source to a given destination. To simplify the notation we consider the indirect Baseline $IB_p = E\sigma_2 E\sigma_3 \dots E\sigma_p E$ and find the position of the message at the ith stage, where the sub-shuffle $\sigma_i(x)$ is defined by $\sigma_i(x_{p-1}, \dots, x_i, x_{i-1}, x_{i-2}, \dots, x_0) = x_{p-1}, \dots, x_i, x_{i-2}, \dots, x_0, x_{i-1}$.

Theorem 4.1 *Given a IB_p network, let $\mathrm{pos}_i(s,d)$ be the position of a message passing from input s to destination d after the ith stage. Then $pos_i(s,d) = s_{p-1} \dots s_i d_{p-1} \dots d_{p-i}$ where d_j is the jth bit of d, and s_j is the jth bit of s.*

Proof The proof is by induction. The base case is $\mathrm{pos}_0(s,d) = s_{p-1} \dots s_0$. We assume that $\mathrm{pos}_i(s,d) = s_{p-1} \dots s_i d_{p-1} \dots d_{p-i}$ and we show that $\mathrm{pos}_{i+1}(s,d) = s_{p-1} \dots s_{i+1} d_{p-1} \dots d_{p-i-1}$ Since the message enters stage $(i+1)$ following a sub-shuffle σ_{i+1}, the position of the message at the entry of stage $(i+1)$ is given by:

$$\sigma_{i+1}(pos_i(s,d)) = s_{p-1} \dots s_{i+1} d_{p-1} \dots d_{p-i} s_i$$

The message exits stage $(i+1)$ at its upper ($d_{p-i-1} = 0$) or lower ($d_{p-i-1} = 1$) output, depending on the routing bit d_{p-i-1}. The position of the message at the output of stage $(i+1)$ is then $pos_{i+1}(s,d) = s_{p-1} \dots s_{i+1} d_{p-1} \dots d_{p-i} d_{p-i-1}$. ∎

4.1 Linear Permutations

Let ϕ be a $p \times p$ *linear permutation matrix* from Z_2^p onto Z_2^p, i.e. $d = \phi s$. We wish to know if the IB_p network can perform ϕ. From Theorem 4.1 we find that:

$$\mathrm{pos}_i(s, \phi(s)) = (\phi_{p-i}(s), \dots, \phi_{p-1}(s), s_i, \dots, s_{p-1})$$

Since we always refer to a message going from s to $\phi(s)$, we shall abbreviate $\mathrm{pos}_i(s, \phi(s))$ to $\mathrm{ps}_i(s)$. If Φ is the matrix of ϕ, the matrix of $\mathrm{ps}_i(s)$ is:

$$\begin{pmatrix} \Phi_{p-i,0} & \cdots & \Phi_{p-i,i-1} & \cdots & \cdots & \Phi_{p-i,p-1} & \\ \vdots & & \vdots & & & & \vdots \\ \Phi_{p-1,0} & \cdots & \Phi_{p-1,i-1} & \cdots & \cdots & \Phi_{p-1,p-1} & \\ 0 & \cdots & 0 & 1 & \cdots & 0 & \\ \vdots & & \vdots & \vdots & & \vdots & \\ 0 & \cdots & 0 & 0 & \cdots & 1 & \end{pmatrix} \tag{1}$$

We define:

$$\Phi[i] = \begin{pmatrix} \Phi_{p-i,0} & \cdots & \Phi_{p-i,i-1} \\ \vdots & & \vdots \\ \Phi_{p-1,0} & \cdots & \Phi_{p-1,i-1} \end{pmatrix} \tag{2}$$

Note that $\Phi[i]$ is just the $i \times i$ sub-matrix in the upper-left corner of matrix (1), which is the $i \times i$ sub-matrix in the lower-left of Φ.

Theorem 4.2 ps_i *is onto if and only if* $\Phi[i]$ *is non-singular.*

Proof ps_i will be onto if and only if its matrix (1) is non-singular. The matrix (1) contains the identity matrix in its lower-right hand corner. We can cancel any entry $\Phi_{j,k} = 1$ where $p-i \leq j \leq p-1$ and $0 \leq k < i$, by adding row k to row j. This leaves the identity in the lower right quadrant, matrix (2) in the upper-left quadrant, and zeros in the remaining quadrants. Therefore, the non-singularity of (1) depends on its upper-left quadrant, which is matrix (2). ■

We now characterize linear permutations ϕ which the inverted baseline network can perform. We say that an $p \times p$ matrix Φ is *sub-non-singular* if and only if $\Phi[i]$ is non-singular for $1 \leq i \leq p-1$.

Theorem 4.3 *A linear permutation* ϕ *on* Z_2^p *can be performed by* IB_p*, if and only if its matrix* Φ *is sub-non-singular.*

Proof If two messages are mapped to the same output of a single switch in the network, the permutation will fail. If some sub-matrix $\Phi[i]$ is singular, then ps_i is not onto, and two messages will be mapped to the same output of some switch in the ith stage. ■

4.2 Sub-Non-Singular Matrices

We find the number of $p \times p$ matrices M defined over Z_2 which are sub-non-singular. Let us denote the number of $p \times p$ matrices which are sub-non-singular by S_p. It is obvious that $S_1 = 1$ and there are $S_2 = 4$ matrices which are sub-non-singular:

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \quad \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$$

Theorem 4.4 *Let* S_p *be the number of sub-non-singular* $p \times p$ *matrices over* Z_2*. Then* $S_p = 2^{(p-1)p}$.

Proof Let M be a $p \times p$ matrix so that $M[i]$ is sub-non-singular. Using row and column operations, we can transform M such that $M[i]$ is the mirror identity matrix:

$$\begin{pmatrix} \vdots & & & & \vdots & \\ b_{i-1} & b_{i-2} & \cdots & b_0 & c & \cdots \\ 0 & 0 & \cdots & 1 & a_0 & \\ \vdots & \vdots & & \vdots & & \\ 1 & 0 & \cdots & 0 & a_{i-1} & \cdots \end{pmatrix}$$

We denote the entry in the upper-right corner as c, the entries below c as $a_0, \ldots, a_{i-1}$, and the entries to the left of c as $b_0, \ldots, b_{i-1}$. By examining these quantities, we can determine whether $M[i+1]$ is non-singular. The fact that $M[i]$ is the mirror-identity makes it easy to cancel the a_i's and b_i's using row and column operations. $M[i+1]$ will be non-singular, if and only if, after these operations $c = 1$. If $a_j = 1$, add the column containing b_j to column $(i-1)$ of M. This changes a_j to zero. If $b_j = 1$ and we add the row containing a_j to row $(p-1-i)$ of M, this changes b_j to zero. These operations affect c as follows: 1) there is no change to c if either $a_j = b_j = 0$ or $a_j \oplus b_j = 1$ and, 2) c is flipped if $a_j = b_j = 1$.

The non-singularity of $M[i+1]$ will therefore depend on two factors: the initial value of c, and the number of flips. There are 3^i ways to get 0 flips because there are three ways each a-b pair can be assigned without causing a flip. There are $i3^{i-1}$ ways we can get one flip, and $i(i-1)3^{i-2}/2$ ways we can get two flips. In general, there are $\binom{i}{j}3^{i-j}$ ways we can get j flips. If c is initially zero, $M[i+1]$ is non-singular exactly when there are an odd number of flips. This can happen in $\sum_{0 \le j \le i, \text{odd} i} \binom{i}{j}3^{i-j}$ different ways. If c is initially one, $M[i+1]$ is non-singular exactly when there are an even number of flips. This happens in $\sum_{0 \le j \le i, \text{even} i} \binom{i}{j}3^{i-j}$ ways. The total number of ways is simply $\sum_{j=0}^{i} \binom{i}{j}3^{i-j}$.

It can be proved [1] that all 4^i values of a's and b's are possible. If there are S_i ways that $M[i]$ can be non-singular, then there are $S_i \sum_{j=0}^{i} \binom{i}{j}3^{i-j} = S_i(3+1)^i = S_i 4^i$ ways that $M[i+1]$ can be non-singular. Combining this with our value for $S_1 = 1$ we have $S_{p+1} = S_p 4^p$. Since S_p is always a power of four, let $s_p = \log_4 S_p$. Then $s_0 = 0$ and $s_{p+1} = s_p + p$. This gives $s_p = \sum_{i=0}^{p-1} i = (p-1)p/2$. Therefore, we have $S_p = 4^{(p-1)p/2} = 2^{(p-1)p}$. ∎

Note that the proof of this theorem implies a $\Theta(p^3)$ algorithm for determining the sub-non-singularity of a matrix. The best known algorithm for determining non-singularity takes more than $\Theta(n^2)$ time. Now we compare the number of non-singular matrices to matrices.

Theorem 4.5 *The number of $p \times p$ matrices that are non-singular is $U_p = \prod_{i=1}^{p}(2^p - 2^{i-1})$.*

Corollary 4.1 *The ratio of $p \times p$ non-singular matrices to matrices is:*

$$V_p = \prod_{i=1}^{p} \frac{2^i - 1}{2^i} = \frac{1}{2} \cdot \frac{3}{4} \cdots \frac{2^p - 1}{2^p} \tag{3}$$

Furthermore, V_p converges as p goes to infinity.

Proof of Theorem 4.5 and corollary 4.1 can be found in [1].

5 XOR-schemes and the Network

Suppose ϕ is an XOR-scheme for a set of templates. ϕ is *network-contention-free* if and only if all template instances of all templates are mapped by ϕ in such a way that the network can perform the mappings.

Theorem 5.1 *Let ϕ be an XOR-scheme for a template set $\mathcal{T} = \{\mathcal{T}_1, \ldots, \mathcal{T}_t\}$, with matrix Φ. Then ϕ is conflict-free and network-contention-free for the Indirect Baseline network if and only if Φ^{T_i} is sub-non-singular for all i.*

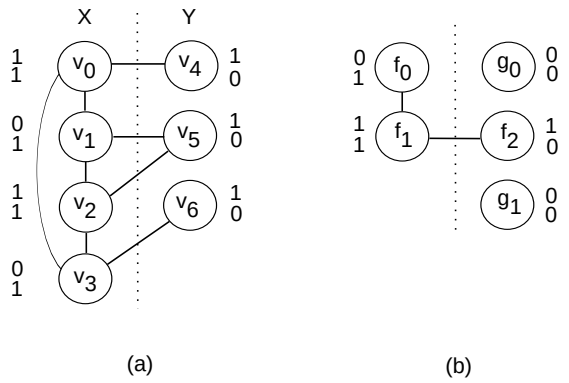

Figure 2: Examples of SNSS with p=2

Proof Consider ϕ restricted to some template $\mathcal{T}_i$. Then this restricted ϕ must be a linear permutation, if it is to be conflict-free. The matrix Φ^{T_i} must therefore be sub-non-singular. So Φ^{T_i} must be sub-non-singular for all i. ∎

5.1 NP-Completeness

We show that the problem of finding an XOR-scheme which allows conflict-free and network-contention-free access for a given set of templates is tractable for $p = 2$, but NP-complete for $p > 2$. We consider the following abstract problem: 1) a vector space $\mathcal{Z} = Z_2^p$, 2) a set of n variables $V = \{v_i \mid 0 \leq i \leq n-1\}$ and, 3) a set T of p-tuples of variables, $T = \{(v_{i_0}, \ldots, v_{i_{p-1}}) \mid 0 \leq i_j \leq n-1\}$. The vectors assigned to the variables of a tuple form the columns of the $p \times p$ matrix. For example, if (v_0, v_1) is a tuple and $v_0 = \binom{0}{1}$ and $v_1 = \binom{1}{1}$, then the matrix of this tuple is $\binom{01}{11}$.

We want to find an assignment of vectors in $\mathcal{Z}$ such that the matrices corresponding to all tuples are sub-non-singular. We call this problem Sub-Non-Singular Satisfaction (SNSS).

This problem can easily be solved for $p = 2$. A 2×2 matrix is sub-non-singular only if its lower left entry is non-zero. Let X be the set of vertices that appear first in some tuples, and thus can be assigned only two values $\binom{0}{1}$ or $\binom{1}{1}$. Let Y be the set of vertices that appear second which can be assigned three values $\binom{0}{1}$, $\binom{1}{0}$ or $\binom{1}{1}$. The remaining vertices can take any values. Two vertices of the same tuple must be assigned different vectors, or the corresponding matrix will be singular. We build a *conflict graph* to this problem. All edges will be between two vertices in X, or between a vertex in X and one in Y. We two-color the vertices in the graph. We call this algorithm XIB2.

Suppose we have variables $v_0, \ldots, v_6$ and $T_1 = (v_0, v_4)$, $T_2 = (v_1, v_5)$, $T_3 = (v_2, v_5)$, $T_4 = (v_3, v_6)$, $T_5 = (v_0, v_1)$, $T_6 = (v_1, v_2)$, $T_7 = (v_2, v_3)$, and $T_8 = (v_0, v_3)$. Sets $X = \{v_0, v_1, v_2, v_3\}$ and $Y = \{v_4, v_5, v_6\}$. The conflict graph of this problem and one possible coloring are shown in Figure 2-a. It is proved [1] that SNSS is NP-hard for p=3. In general, SNSS corresponds to 2^{p-1}-coloring which is an NP-complete problem for $p \geq 3$.

6 Heuristic Approach to SNSS

We present an algorithm for finding an XOR-scheme for a given template set. The idea is to construct the matrix of the XOR-scheme one row at a time, from the bottom up. We use algorithm XIB2 along with the proof of Theorem 4.4. Suppose $p = 3$ and we are given the templates shown in Figure 1. The bases of these templates are given in Section 3.

We construct the bottom two rows of Φ using algorithm XIB2. For each Φ^{T_i}, the lower-left 2×2 sub-matrix is to be sub-non-singular. The reduced template bases are $T'_1 = T'_2 = T'_4 = \{f_0, f_1\}$ and $T'_3 = \{f_1, f_2\}$. The sets of vectors that appear first and second in some template bases are $X = \{f_0, f_1\}$ and $Y = \{f_2\}$, respectively. The conflict graph and one possible coloring are shown in Figure 2-b. We let $x_0, \ldots, x_4$ be the values in the top row:

$$\Phi = \begin{array}{c} \begin{array}{ccccc} f_0 & f_1 & f_2 & g_0 & g_1 \end{array} \\ \begin{pmatrix} x_0 & x_1 & x_2 & x_3 & x_4 \\ 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

We must now ensure that each Φ^{T_i} is non-singular. The first step is to get the mirror identity matrix in the lower right 2×2 sub-matrix, using only row operations. Using the notation of Theorem 4.4, each Φ^{T_i} is non-singular if $c = 1$ and the number of pairs $a_i = b_i = 1$ is even, or if $c = 0$ and the number of pairs $a_i = b_i = 1$ is odd. We can express this as linear equations over Z_2 which gives $x_0 \oplus x_1 \oplus x_2 = 1$ for Φ^{T_1}, $x_4 = 1$ for Φ^{T_2}, and $x_3 = 1$ for Φ^{T_3} and Φ^{T_4}. One solution for this system of simultaneous equations is $x_1 = x_3 = x_4 = 1$ and $x_1 = x_2 = 0$. The final storage matrix is:

$$\Phi = \begin{array}{c} \begin{array}{ccccc} f_0 & f_1 & f_2 & g_0 & g_1 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \end{pmatrix} \end{array} \tag{4}$$

The permutations corresponding to Φ^{T_1}, Φ^{T_2}, and Φ^{T_3} map $(0, 1, \ldots, 7)$ into $(0, 2, 3, 1, 5, 7, 6, 4)$, $(0, 4, 3, 7, 5, 1, 6, 2)$, and $(0, 4, 2, 6, 3, 7, 1, 5)$, respectively. Note that $\Phi^{T_2} = \Phi^{T_4}$. All these permutations are memory and network conflict-free with respect to an IB_3 network.

In general, the algorithm for finding a SNSS storage matrix is:

1. Determine the bottom two rows of the matrix using algorithm XIB2.
2. Create each remaining row, working from the bottom up.
   For i in 2 to $p - 1$ loop:
   (a) For each template $\mathcal{T}_j$ do:
      i. Obtain a matrix $\hat{\Phi}^{T_j}$ by reducing the matrix Φ^{T_j} so that it has the mirror identity matrix in its lower-left corner, using only row operations. Operations do not affect the matrix Φ.

ii. Use the ith column of this matrix to determine the equation associated with this template. Let the basis of $\mathcal{T}_j$ be $v_{\ell_0}, \ldots, v_{\ell_{p-1}}$, and $y_k = \hat{\Phi}^{T_j}_{p-k-1,\ell_i}$. Then the equation is:

$$x_{\ell_i} \oplus \bigoplus_{k=0}^{i-1} x_{\ell_k} y_k = 1 \tag{5}$$

(b) Solve the system of simultaneous equations. Assign entry $\Phi_{p-i-1,k}$ the value x_k.

We call this algorithm XIB. Note that we do not have to row reduce from scratch each time we perform Step i. We can directly row reduce this partially reduced matrix and reduce the time complexity of this step from $O(p^3)$ to $O(p^2)$. The complexity of algorithm XIB is $O(ptn^2)$, where t, 2^p, and n, are the number of templates, the number of processors, and the number of distinct vectors of the template bases, respectively.

6.1 Approximate Solutions

Algorithm XIB fails if the sub-graph X cannot be two-colored. To find approximate two-coloring we assume that each template has a weight of 1. The problem is to find a two-coloring of X which violates the least number of edges. Garey, Johnson, and Stockmeyer [8] have shown this problem to be NP-complete. Algorithm XIB fails if a solution to the set of equations cannot be found. The problem is to find an *Approximate Linear Solution* (ALS) to the equations, for which the sum of the weights is minimized.

Theorem 6.1 *Approximate Linear Solution is NP-hard.*

Proof We can use an algorithm for ALS to solve the problem of finding an optimal approximate two-coloring, which is NP-complete [8]. Suppose we are given an arbitrary graph G, and we wish to find an approximate two-coloring which violates the minimum number of edges. In Z_2, for each vertex v_i of G we create a variable v_i. For each edge (v_i, v_j), we create an equation $v_i \oplus v_j = 1$ and give it weight 1. It is easily seen that an optimal approximate solution to the resulting system of equations corresponds directly to an optimal approximate two-coloring of G. ∎

If contention occurs at one stage for a given Φ^T, then the inputs of each switch of that stage must be serialized, i.e. its cost is two. If contention occurs at c stages the cost is 2^c. Contention will occur at stage i if and only if the $rank(\Phi^T[i]) = rank(\Phi^T[i-1])$. We define $\text{rank}(\Phi^T[0]) = 0$. Let $C_i = 1$ if $\text{rank}(\Phi^T[i]) > \text{rank}(\Phi^T[i-1])$, and $C_i = 0$ otherwise. The *subrank* of Φ^T is then $\text{subrank}(M) = \sum_{i=1}^{p} C_i$. The cost of an XOR-scheme Φ will be:

$$\text{cost}(\Phi) = \sum_{i=1}^{t} w_i 2^{p - \text{subrank}(\Phi^{T_i})} \tag{6}$$

In particular, the cost of an XOR-scheme that is network-contention-free will be $\sum w_i$. Algorithm XIB is performed repeatedly, until an XOR scheme within pre-set performance parameters is found, or an iteration limit is reached. In the later case, the best XOR-scheme found is used. We call this randomized algorithm RXIB.

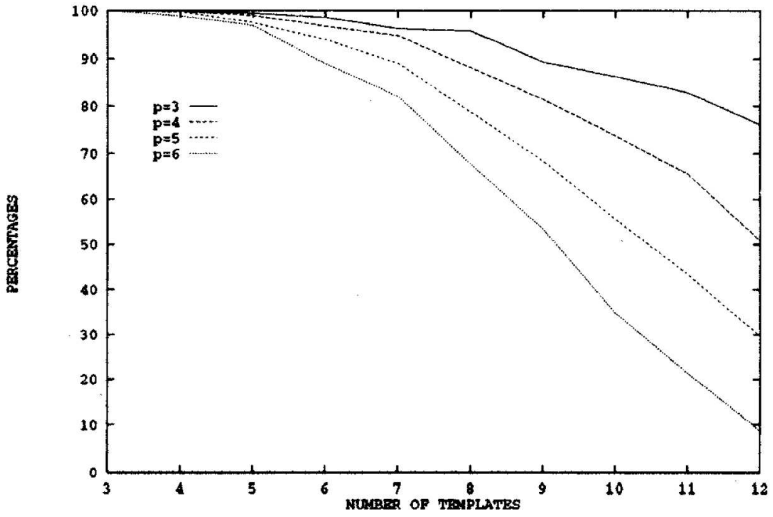

Figure 3: Percentage of optimum cases found

7 Evaluation

Testing RXIB is performed as follows. The number of templates t ranges from 3 to 12, the weight was set to 1, and p ranges from 3 to 6, where 2^p is the number of processors. One thousand cases were generated for each combination of these parameters.

For each value of t and p, the left and right entries of Figure 3 and 4 show 1) the percentage of cases where an optimum solution was found and, 2) the average time increasing over the optimum access time. Our scheme finds near optimum solutions for small numbers of templates and moderate numbers of processors. For the cases where the optimum solution is not found, the average deviation from the optimum access time is moderate in all studied cases. The degradation smoothly increases with increasing the number of templates or the number of processors.

Algorithm XIB found a memory and network-contention free scheme when the set of templates is formed by power-of-2 strides. We compare RXIB to row-major interleaving (INT) and to a static-storage-scheme (SSS) that consists of the row, column, and both diagonals [5]. In the case of arbitrary templates, the INT scheme causes the average access time to be 6, 9.37, 13.59, and 18.64 fold the optimum access time for 2^3, 2^4, 2^5, and 2^6 processors, respectively. Similarly, the SSS scheme causes the average access time to be 4.23, 5.31, 5.79, and 5.84 fold the optimum access time for the same number of processors. RXIB significantly outperforms both INT and SSS under the studied conditions.

In the following we compare with other approaches. In [7], conflict-free access to parallel memories based on full-rank matrix transformations was proposed. However, the network aspects were not considered. Also, no method was proposed for finding the XOR-scheme in the case of composite templates. The idea of using non-singular matrix transformation has also been reported in [14] for vector processors. We proposed an efficient approach for combining data templates into a single storage matrix for which the condition to conflict-free access can be easily formulated. While it is simple to find the XOR-matrix of one template, we proved that finding a combined XOR-scheme is an NP-complete problem.

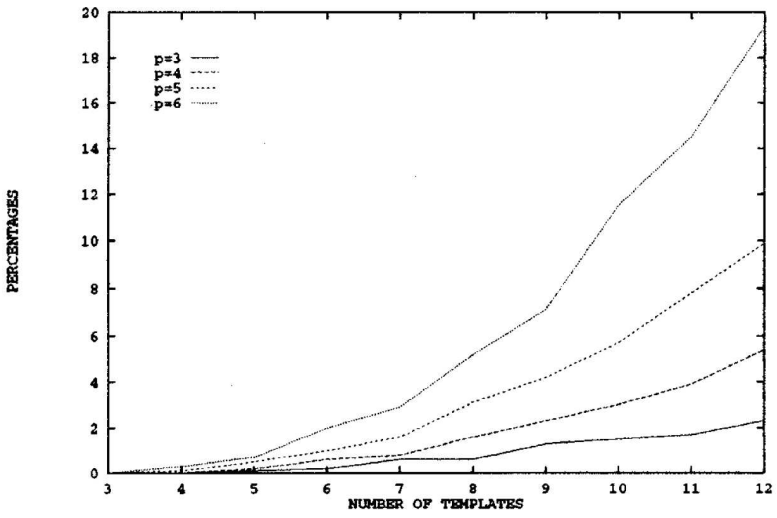

Figure 4: Percentage of time increasing over optimum access time

Where the network aspects are considered, the problem is restricted to finding a storage scheme for a well defined set of templates [5, 12]. For example, minimizing memory and network contention for a subset of rows, columns, diagonals, and square blocks was proposed in [5]. In [12], network contention has been analyzed with respect to conflict-free access to a fixed set of strides. Our method has the advantage of being a general approach that incorporates the constraints of conflict-free access to arbitrary sets of templates with respect to memory and network.

Considering other networks such as *Omega*, *Cube*, and *Delta*, the position of the message at some stage can always be expressed as a combination of bits of the source and destination. Therefore, the results presented here are also applicable to other multistage networks.

8 Conclusion

We investigated the problem of finding general XOR-schemes to dynamically minimize memory and network contention in accessing arrays with arbitrary data templates in SIMD computers.

Characterization of linear permutations for the Baseline network was presented by using non-singular boolean matrices which guarantee conflict-free access to both memory and networks. We proved that finding the XOR-matrix for accessing arbitrary data templates is an NP-complete problem. To minimize memory and network contention, a heuristic algorithm was proposed for finding storage schemes for accessing an arbitrary set of data templates. Evaluation shows that the proposed XOR-schemes significantly reduce the memory and network contention compared to interleaving and other static storages.

The contributions of this work are: 1) a general approach for finding combined storage schemes, 2) characterization of necessary and sufficient conditions for conflict free-access of memory and network and, 3) an efficient algorithm for automating the process of finding the combined XOR-matrix.

9 acknowledgment

Thanks to the College of Computer Science and Engineering, King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia, for granting the sabbatical leave of the first author. Thanks to Professors Daniel Hirschberg and Lubomir Bic, Department of Information and Computer Science, University of California, Irvine, for listening critically to the various proofs and for their remarks concerning the presentation of this paper.

References

[1] M. Al-Mouhamed and S. Seiden. Minimization of memory and network contention for accessing arbitrary data patterns in SIMD systems. *University of California Irvine, ICS-UCI Technical report 93-29*, Jun 1993.

[2] D. Bailey. Vector computer memory bank contentions. *IEEE Trans. on Computers*, C-36:293–298, Mar 1987.

[3] K. Batcher. The multidimensional access memory in STARAN. *IEEE Trans. on Computers*, C-26:174–177, Feb 1977.

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

[5] R. V. Boppana and C. S. Raghavendra. Efficient storage schemes for arbitrary size square matrices in parallel processors with shuffle-exchange networks. In *Proceedings of the International Conference on Parallel Processing*, pages 365–368, 1991.

[6] P. Budnik and D. Kuck. The organization and use of parallel memories. *IEEE Trans. on Computers*, C-20, No 12:1566–1569, Dec 1971.

[7] J. M. Jalby W. Frailong and J. Lenfant. XOR-schemes: A flexible data organization in parallel memories. In *Proceedings of the International Conference on Parallel Processing*, pages 276–283, 1985.

[8] M. R. Garey, D. S. Johnson, and Stockmeyer L. Some simplified NP-complete graph problems. *Theoretical Computer Science*, 2:237–267, 1976.

[9] D. T. Harper III. Block, multistride vector, and FFT accesses in parallel memory systems. *IEEE Trans. on Parallel and Distributed Systems*, 2, No 1:43–51, Jan 1991.

[10] D. Lawrie. Access and alignment of data in an array processor. *IEEE Trans. on Computers*, C-24, No 12:1145–1155, Dec 1975.

[11] D. Lawrie and C.R. Vora. The prime memory system for array accesses. *IEEE Trans. on Computers*, C-31, 12:435–442, May 1982.

[12] A. Norton and E. Melton. A class of boolean linear transformations for conflict-free power-of-two stride access. *Proceedings of the International Conference on Parallel Processing*, pages 247–254, 1987.

[13] H. J. Siegel. Interconnection networks for SIMD machines. *Computer*, 12:57–67, Jun 1979.

[14] G. S. Sohi. High-bandwidth interleaved memories for vector processors–A simulation study. *IEEE Trans. on Computers*, 42, No 1:34–44, Jan 1993.