SIAM J. COMPUT.
Vol. 15, No. 1, February 1986

© 1986 Society for Industrial and Applied Mathematics

AN EFFICIENT ALGORITHM FOR GENERATING LINEAR TRANSFORMATIONS IN A SHUFFLE-EXCHANGE NETWORK*

T. ETZION† AND A. LEMPEL†

Abstract. This paper presents an algorithm for generating all the permutations defined by linear transformations on a shuffle-exchange network of 2^n processors in $2n-1$ passes. The proposed algorithm generates any such permutation in $O(n \log^2 n)$ elementary steps. The subclass of bit-permutations is generated in $O(n)$ steps.

Key words. algorithm, complexity, linear transformations, permutations, shuffle-exchange network

1. Introduction. The shuffle-exchange (SE) network is an efficient tool for implementing various types of parallel processes [2], [6]. The SE network is composed of $N=2^n$ processors, where each processor is represented by a binary n-tuple $(x_1, x_2, \cdots, x_n)$. In the SHUFFLE-operation processor $(x_1, x_2, \cdots, x_n)$ transfers information to processor $(x_2, \cdots, x_n, x_1)$. In the EXCHANGE-operation processors $(x_1, x_2, \cdots, x_{n-1}, 0)$ and $(x_1, x_2, \cdots, x_{n-1}, 1)$ may exchange information, independent of other pairs of this form.

One SHUFFLE followed by one EXCHANGE is called a *pass*. Between the SHUFFLE phase and the EXCHANGE phase of a pass there is a computational phase during which the active pairs of the upcoming EXCHANGE are determined. Prior to the first pass there is normally a preprocessing stage. The overall procedure consisting of the preprocessing stage and all the passes is often referred to as the *routing algorithm.*

An important problem in this context is the design of efficient routing algorithms that implement permutations in a SE network in a minimal number of passes. In general, a transformation on a SE network associates with each processor a destination processor for the purpose of information transfer. This paper deals with the realization of nonsingular linear transformations, i.e., permutations for which each bit of the destination processor is a linear combination of the bits of the origin processor. It is well known ([3] and [4]) that such permutations can be realized in $2n-1$ passes, using a routing algorithm of $O(n^2)$ steps.

In § 2 we show how to realize these permutations in $2n-1$ passes, using a routing algorithm of $O(n \log^2 n)$ steps. In § 3 we show that if the permutation is merely a bit permutation, then only $O(n)$ steps are required.

Following Linial and Tarsi [3], we employ the combinatorial model described below.

DEFINITION 1[3]. A 0-1 matrix $\mathbf{A}$, of order $N \times m$, $N=2^n$, $m \geqq n$, is said to be *balanced* if all the rows in any n consecutive columns of $\mathbf{A}$ are distinct.

DEFINITION 2. The *standard* matrix is an $N \times n$ matrix $\mathbf{D}$ whose ith row is the base-2 representation of i, $0 \leqq i \leqq N-1$.

In terms of these definitions our problem can be stated as follows [3]: Given a balanced $N \times n$ matrix $\mathbf{A}$ find a (possibly empty) matrix $\mathbf{X}$ such that the matrix $[\mathbf{D} \mid \mathbf{X} \mid \mathbf{A}]$ is balanced.

2. Realization of linear transformations. In this section we show how to realize linear transformations on a SE network in $2n-1$ passes using a routing algorithm of $O(n \log^2 n)$ steps.

* Received by the editors May 9, 1984, and in final form November 9, 1984.

† Computer Science Department, Technion, Israel Institute of Technology, Haifa, Israel.

In the sequel all the arithmetic is over $GF(2)$, all the vectors are column vectors with n elements, $\mathbf{I}=[I(1)I(2)\cdots I(n)]$ denotes the identity matrix of order n, and $\mathbf{T}$ denotes a nonsingular matrix of order n.

PROPOSITION 1[3]. *Let* $\mathbf{A}$ *be a matrix of order* $N\times n$. *Then* $\mathbf{AT}$ *is balanced if and only if* $\mathbf{A}$ *is.*

DEFINITION 3. A matrix $\mathbf{R}$ of order $n\times m$, $n\leqq m$, is said to be *n-regular* if every n consecutive columns of $\mathbf{R}$ are linearly independent.

It follows readily from Proposition 1 that if $\mathbf{R}$ is n-regular then $\mathbf{DR}$ is balanced. In what follows we consider n-regular matrices of the form $\mathbf{R}=[\mathbf{I}\,|\,\mathbf{Y}\,|\,\mathbf{T}]$ and propose a method of finding a suitable matrix $\mathbf{Y}$ of $n-1$ columns when given the matrix $\mathbf{T}$.

Consider a $n\times n$ binary matrix $\mathbf{B}=[B(1)B(2)\cdots B(n)]$, where each column $B(i)$ has either one or two nonzero entries. $\mathbf{B}$ can be viewed as the incidence matrix of the (undirected) graph $G(\mathbf{B})$ defined as follows:

DEFINITION 4. $G(\mathbf{B})$ has $n+1$ vertices $0,1,\cdots,n$ and n edges $e(1), e(2),\cdots,e(n)$, where $e(k)$ joins vertices $i>0$ and $j>0$ if $B(k)$ has nonzero entries in rows i and j, and $e(k)$ joins vertices $i>0$ and 0 if $B(k)$ is nonzero in row i only.

LEMMA 1[5]. *The vectors* $B(1), B(2),\cdots,B(n)$ *are linearly independent if and only if* $G(B)$ *is a tree.*

LEMMA 2. *Let* $B(1)$, $B(2),\cdots,B(n)$ *be linearly independent vectors. Then there exists an integer* k, $1\leqq k\leqq n$, *and binary coefficients* b_j, $1\leqq j\leqq n-1$, *such that*

$$I(k)=B(n)+\sum_{j=1}^{n-1} b_jB(j). \tag{1}$$

Proof. The matrix $\mathbf{B}=[B(1)B(2)\cdots B(n)]$ is nonsingular. Hence, there exists a matrix $\mathbf{Q}=[Q(1)Q(2)\cdots Q(n)]$ such that $\mathbf{BQ}=\mathbf{I}$. Since $\mathbf{Q}$ is nonsingular, there exists at least one k such that the last entry of $Q(k)$ equals 1. Q.E.D.

LEMMA 3. *Let* $B(1)$, $B(2),\cdots,B(n)$ *be linearly independent vectors and let* k, $1\leqq k\leqq n$ *be an integer satisfying Lemma* 2. *Then* $B(0)$, $B(1)$, $B(2),\cdots,B(n-1)$ *are linearly independent, where*

$$B(0)=I(k)+\sum_{j=1}^{n-1} c_jB(j), \qquad c_j\in\{0,1\}. \tag{2}$$

Proof. Assume, the contrary, that $B(0)$, $B(1)$, $B(2),\cdots,B(n-1)$ are linearly dependent. Then, since the last $n-1$ vectors are linearly independent, there exist d_j, $1\leqq j\leqq n-1$, such that

$$B(0)=\sum_{j=1}^{n-1} d_jB(j). \tag{3}$$

From (1), (2), and (3), we obtain

$$B(n)=\sum_{j=1}^{n-1}(b_j+c_j+d_j)B(j)$$

which contradicts the linear independence of the $B(j)$, $1\leqq j\leqq n$. Q.E.D.

Based on Lemmas 1 and 3, we propose the following construction of $\mathbf{Y}=[Y(1)Y(2)\cdots Y(n-1)]$ such that $[\mathbf{I}\,|\,\mathbf{Y}\,|\,\mathbf{T}]$ be n-regular for a given $\mathbf{T}=[T(1)T(2)\cdots T(n)]$.

Construction 1. Let $\mathbf{B}_0=\mathbf{T}$ and let $\mathbf{B}_m=[Y(n-m)\cdots Y(n-1)T(1)\cdots T(n-m)]$, $1\leqq m\leqq n-1$. Given $\mathbf{B}_m$, $0\leqq m<n-1$, construct $Y(n-m-1)$ as follows.

(i) If $k=n-m-1$ satisfies Lemma 2, set $Y(n-m-1)=I(n-m-1)$.

(ii) If $k=n-m-1$ does not satisfy Lemma 2, find an integer q which does satisfy Lemma 2 and set $Y(n-m-1)=I(n-m-1)+I(q)$.

LEMMA 4. *The matrix* $[I(1)\cdots I(n)Y(1)\cdots Y(n-1)T(1)\cdots T(n)]$ *obtained via Construction* 1 *is n-regular.*

Proof. The n-regularity of $[Y(1)\cdots Y(n-1)T(1)\cdots T(n)]$ follows directly from Lemma 3. To complete the proof, it suffices to show that $[J(1)\cdots I(n)Y(1)\cdots Y(n-1)]$ is n-regular. Let $\mathbf{C}_1=\mathbf{I}$ and let $\mathbf{C}_m=[I(m)\cdots I(n)Y(1)\cdots Y(m-1)]$, $1<m\leqq n$. We will show that linear independence among the columns of $\mathbf{C}_m$, $1\leqq m<n$ implies the same for $\mathbf{C}_{m+1}$. Clearly, the columns of $\mathbf{C}_1$ are linearly independent. Suppose $\mathbf{C}_m$, $m\geqq 1$, is nonsingular and consider $\mathbf{C}_{m+1}=[I(m+1)\cdots I(n)Y(1)\cdots Y(m)]$. By Construction 1, either $Y(m)=I(m)$ or $Y(m)=I(m)+I(q)$ for some $q\neq m$. In the first case it is clear that $\mathbf{C}_{m+1}$ is nonsingular. In the latter case noting that $\mathbf{C}_r$, $1\leqq r<n$, has at most two nonzero entries in every column, we can view $\mathbf{C}_r$ as the incidence matrix of the graph $G(\mathbf{C}_r)$ according to Definition 4. By Lemma 1, since $\mathbf{C}_m$ is nonsingular $G(\mathbf{C}_m)$ is a tree. $G(\mathbf{C}_{m+1})$ is obtained from $G(\mathbf{C}_m)$ by deleting the edge $(0, m)$ (corresponding to the column $I(m)$) and inserting the edge (m, q) (corresponding to the column $I(m)+I(q)$). If $G(\mathbf{C}_{m+1})$ contains a cycle then, since $Y(1),\cdots, Y(n-1)$ are linearly independent, the cycle must include the vertex 0. Since $G(\mathbf{C}_m)$ is a tree, deleting the edge $(0, m)$ from $G(\mathbf{C}_m)$ leaves a graph with no path between the vertices 0 and m. Hence, inserting the edge (m, q) cannot generate a cycle that contains the vertex 0. Hence $G(\mathbf{C}_{m+1})$ is a tree, and $\mathbf{C}_{m+1}$ is nonsingular. Q.E.D.

In order to find an integer k that satisfies Lemma 2, we need an efficient algorithm to invert a matrix. To this end we use the algorithm proposed by Csanky [1] who showed how to invert a matrix of order n, in $O(\log^2 n)$ steps using a polynomial number of processors. Csanky's algorithm utilizes a model that has an arbitrary number of identical processors with independent control and an arbitrarily large shared memory with unrestricted access. In this model, each processor is capable of taking its operands from the shared memory performing any one of the binary operations $+, -, *, /$ and storing the result in memory in one step.

Based on Lemma 2, Csanky's algorithm, and Construction 1, we propose a procedure to realize linear transformations. In this procedure each processor has at each stage the following information:

(1) An $(n-1)$-tuple $U=(u(1),\cdots,u(n-1))$, where $u(j)=k$ if $Y(j)=I(j)+I(k)$ and $u(j)=0$ if $Y(j)=I(j)$.

(2) Two n-tuples, S and $F=S\mathbf{T}$, whose initial values represent, respectively, the ID of the said processor and that of the destination processor as defined by the given linear transformation. In the SHUFFLE and the EXCHANGE operations that follow each processor transfers its current S and F and receives new values for S and F.

Procedure 1. Given the linear transformation defined by a nonsingular matrix $\mathbf{T}=[T(1)T(2)\cdots T(n)]$ let $\mathbf{B}_0=T$ and let $\mathbf{B}_m=[Y(n-m)\cdots Y(n-1)T(1)\cdots T(n-m)]$. Having computed $\mathbf{B}_r=[Y(n-r)\cdots Y(n-1)T(1)\cdots T(n-r)]$, $r\geqq 0$, apply the Csanky algorithm to generate the inverse $\mathbf{Q}=[Q(1)\cdots Q(n)]$ of $\mathbf{B}_r$. If the last entry of $Q(n-1)$ equals 1, set $Y(n-r-1)=I(n-r-1)$ and $u(n-r-1)=0$; otherwise, find an integer k such that the last entry of $Q(k)$ equals 1 and set $Y(n-r-1)=I(n-r-1)+I(k)$ and $u(n-r-1)=k$. After $u(1),\cdots,u(n-1)$ are generated they are transferred to each of the N processors of the SE network. With reference to the n-tuples $S=(s(1),\cdots,s(n))$ and $F=(f(1),\cdots,f(n))$, stored with each processor, perform the following:

```
$s(0) := 0$;
for $i := 1$ to $n-1$ do
begin
  SHUFFLE;
  if $s(u(i)) \neq 0$ then EXCHANGE;
end;
SHUFFLE;
if $s(n) \neq f(1)$ then EXCHANGE;
for $i := 1$ to $n-1$ do
begin
  SHUFFLE;
  if $f(i+1) \neq s(i) + s(u(i))$ then EXCHANGE;
end;
```

Note that SHUFFLE and EXCHANGE are executed in parallel by all the processors.

THEOREM 1. *Procedure* 1 *realizes a linear transformation in* $2n-1$ *passes using a routing algorithm of* $O(n \log^2 n)$ *steps.*

Proof. In order to show that Procedure 1 realizes the linear transformation associated with the matrix $\mathbf{T}$, it suffices to show that it implements the moves implied by the balanced matrix $\mathbf{DB} = \mathbf{D}[\mathbf{I} \mid Y(1) \cdots Y(n-1) \mid \mathbf{T}] = [\mathbf{D} \mid \mathbf{D}Y(1) \cdots \mathbf{D}Y(n-1) \mid \mathbf{DT}]$. That is, for a given processor $S = (s(1), \cdots, s(n))$ and its destination processor $F = (f(1), \cdots, f(n))$, the path in the SE network via which the transformation $F = S\mathbf{T}$ is implemented by Procedure 1 is given by the sequence of processors corresponding to successive n-tuples from the row $S\mathbf{B} = s(1), \cdots, s(n), (s(1) + s(u(1))), (s(2) + s(u(2))), \cdots, (s(n-1) + s(u(n-1))), f(1), \cdots, f(n)$. To this end, note that for each row $S\mathbf{B}$, Procedure 1 performs an EXCHANGE if and only if the leading bit of the current processor differs from the last bit of the succeeding processor.

The claimed complexity of Procedure 1 is obtained as follows. The n-regular matrix $\mathbf{B} = [\mathbf{I} \mid Y(1) \cdots Y(n-1) \mid \mathbf{T}]$ is generated by $n-1$ applications of the Csanky algorithm. Therefore, this part consists of $O(n \log^2 n)$ steps. The $(n-1)$-tuple $U = (u(1), \cdots, u(n-1))$ is transferred to each of the N processors of the SE network on a bus in $O(n)$ steps. The $2n-1$ passes correspond to last $2n-1$ columns of $\mathbf{DB}$ and each pass is executed in constant time. Thus, the overall complexity of the procedure is $O(n \log^2 n)$. Q.E.D.

3. Realization of bit-permutations.

In this section we show how to realize the linear transformation associated with a permutation matrix $\mathbf{T}$ in $O(n)$ steps.

DEFINITION 5. $\mathbf{T} = [T(1)T(2) \cdots T(n)]$ is called a permutation matrix if $T(j) = I(p(j)), j = 1, 2, \cdots, n$, where $p(1), p(2), \cdots, p(n)$ is an arbitrary permutation on the integers $1, 2, \cdots, n$.

Based on Lemma 1, we propose the following construction of $\mathbf{Y} = [Y(1)Y(2) \cdots Y(n-1)]$ such that $[\mathbf{I} \mid \mathbf{Y} \mid \mathbf{T}]$ is n-regular for a given permutation matrix $\mathbf{T} = [I(p(1))I(p(2)) \cdots I(p(n))]$.

Construction 2. Let $\mathbf{B}_0 = \mathbf{T}$ and let $\mathbf{B}_m = [Y(n-m) \cdots Y(n-1)I(p(1)) \cdots I(p(n-m))]$, $1 \leqq m \leqq n-1$. Along with the columns of $\mathbf{Y}$ we construct a sequence of graphs G_i, $0 \leqq i \leqq n-1$. G_0 is the edgeless graph of n isolated vertices $1, 2, \cdots, n$, and given $\mathbf{B}_m$ and G_m, $0 \leqq m < n-1$, construct $Y(n-m-1)$ and G_{m+1} as follows.

If the addition of edge $(n-m-1, p(n-m))$ to G_m creates a cycle, set $Y(n-m-1) = I(n-m-1)$ and $G_{m+1} = G_m$; otherwise, set $Y(n-m-1) = I(n-m-1) + I(p(n-m))$ and obtain G_{m+1} by adding the edge $(n-m-1, p(n-m))$ to G_m.

LEMMA 5. *The matrix* $[I(1) \cdots I(n) Y(1) \cdots Y(n-1) I(p(1)) \cdots I(p(n))]$ *obtained via Construction* 2 *is n-regular.*

Proof. First, observe that every column of the matrix $\mathbf{B}_r$, $0 \leqq r \leqq n-1$, has at most two nonzero entries and, thus, can be viewed as the incidence matrix of the graph $G(\mathbf{B}_r)$ defined in § 2. Note that G_r, as defined by Construction 2, can be obtained from $G(\mathbf{B}_r)$ by deleting from the latter the vertex 0 and all the edges incident with this vertex. Note further that $G(\mathbf{B}_{m+1})$ is obtained from $G(\mathbf{B}_m)$ by:

(i) Deletion of edge $(0, p(n-m))$.

(ii) Addition of either edge $(0, n-m-1)$, or edge $(p(n-m), n-m-1)$.

Assume that $G(\mathbf{B}_m)$, $m \geqq 0$, is a tree. Then, operation (i) results in two pieces of $G(\mathbf{B}_m)$, with no path between vertices 0 and $P(n-m)$. Hence, if at this stage connecting vertex $p(n-m)$ to vertex $n-m-1$ creates a cycle, it follows that operation (i) leaves vertex $n-m-1$ in the same piece with vertex $p(n-m)$, namely with no path between vertex 0 and vertex $n-m-1$. Therefore, in this case, the graph $G(\mathbf{B}_{m+1})$ obtained in operation (ii) by adding the edge $(0, n-m-1)$ is a tree.

If, on the other hand, connecting vertex $p(n-m)$ to vertex $n-m-1$, after operation (i), does not create a cycle in the piece containing vertex $p(n-m)$, it certainly does not create a cycle with vertex 0 and the resulting graph is again a tree.

Since $G(\mathbf{B}_0)$ is a tree it follows that $G(\mathbf{B}_m)$ is a tree for all $0 \leqq m \leqq n-1$, which implies that the matrix $[Y(1) \cdots Y(n-1) I(p(1)) \cdots I(p(n))]$ is n-regular.

The n-regularity of the matrix $[I(1) \cdots I(n) Y(1) \cdots Y(n-1)]$ follows in the same manner as in the proof of Lemma 4. Q.E.D.

The reader can readily verify that the result of Construction 2 could be obtained via Construction 1 through an appropriate choice of the parameter k of Lemma 2.

Construction 2 leads to Procedure 2, given below for realizing bit-permutations. In this procedure, which is simpler than Procedure 1, each processor has at each stage the following information:

(1) an $(n-1)$-tuple $U = (u(1), \cdots, u(n-1))$ as in Procedure 1;

(2) an n-tuple S as in Procedure 1;

(3) the permutation $P = (p(1), \cdots, p(n))$.

```
Procedure 2.
  Part 1
  for $i := 1$ to $n-1$ do
  begin
    $u(i) := p(i+1)$;
    check $(i) :=$ false;
  end;
  check $(n) :=$ false;
  for $i := 1$ to $n-1$ do
  begin
    cycle := false;
    current := $i$;
    while ((cycle = false) and (current $< n$) and (check(current) = false)) do
    begin
      check (current) := true;
      if $u$(current) $\neq i$ then current := $u$ (current)
      else cycle := true;
    end;
    if cycle = true then $u(i) := 0$;
  end;
```

```
Part 2
$s(0) := 0$;
for $i := 1$ to $n-1$ do
begin
  SHUFFLE;
  if $s(u(i)) \neq 0$ then EXCHANGE;
end;
SHUFFLE;
if $s(n) \neq s(p(1))$ then EXCHANGE;
for $i := 1$ to $n-1$ do
begin
  SHUFFLE;
  if $s(p(i+1)) \neq s(i) + s(u(i))$ then EXCHANGE;
end;
```

THEOREM 2. *Procedure 2 realizes a bit-permutation in $2n-1$ passes and $O(n)$ steps.*

Proof. In Part 1 of Procedure 2 each processor computes the $(n-1)$-tuple $U = (u(1), \cdots, u(n-1)$. Initially $u(n-m-1)$ is set to $p(n-m)$ which corresponds to setting $Y(n-m-1)$ to $I(n-m-1)+I(p(n-m))$. Then, $u(n-m)$ is set to 0 if the insertion of the edge $(n-m-1, p(n-m))$ creates a cycle in the corresponding graph G_m. Part 2 of Procedure 2 is identical to Procedure 1, with $s(p(i))$ substituting for $f(i)$.

The claimed complexity of Procedure 2 is obtained as follows. Part 1 consists of $O(n)$ steps since the variables *check*(i), $1 \leqq i \leqq n$, insure that for each i, the variable *current* takes the value i at most once in the while loop. As in Procedure 1, each of the $2n-1$ passes is executed in constant time. Thus, the overall complexity of the procedure is $O(n)$. Q.E.D.

Acknowledgment. The authors wish to thank Yossi Shiloach for presenting the problem and for many helpful discussions.

REFERENCES

[1] L. CSANKY, *Fast parallel matrix inversion algorithms*, this Journal, 4 (1976), pp. 618–623.

[2] D. H. LAWRIE, *Access and alignment of data in an array processor*, IEEE Trans. Comput., C-25 (1976), pp. 55–65.

[3] N. LINIAL AND M. TARSI, *Efficient generation of permutations with the shuffle exchange networks*, submitted for publication.

[4] M. PEASE, *The indirect binary n-cube microprocessor*, IEEE Trans. Comput., C-26 (1977), pp. 122–131.

[5] S. SESHU AND M. B. REED, *Linear Graphs and Electrical Networks*, Addison-Wesley, Reading, MA, 1961.

[6] H. S. STONE, *Parallel processing with the perfect shuffle*, IEEE Trans. Comput., C-20 (1971), pp. 153–161.