A DIRECT ROUTING ALGORITHM FOR THE BIT-REVERSAL PERMUTATION ON A SHUFFLE-EXCHANGE NETWORK.

Ingrid J.M. Birkhoff

Technical Report RUU-CS-84-3
februari 1984

Rijksuniversiteit Utrecht

Vakgroep informatica

Budapestlaan 6 3584 CD Utrecht
Corr. adres: Postbus 80.012 3508 TA Utrecht
Telefoon 030-531454
The Netherlands

A DIRECT ROUTING ALGORITHM FOR THE BIT-REVERSAL PERMUTATION ON A SHUFFLE-EXCHANGE NETWORK.

Ingrid J.M. Birkhoff

Technical Report RUU-CS-84-3
februari 1984

Department of Computer Science
University of Utrecht
P.O. Box 80.012, 3508 TA Utrecht
the Netherlands

A DIRECT ROUTING ALGORITHM FOR THE BIT-REVERSAL PERMUTATION ON A SHUFFLE-EXCHANGE NETWORK.

Ingrid J.M. Birkhoff

Department of Computer Science, University of Utrecht
P.O. Box 80.012, 3508 TA Utrecht, the Netherlands

Abstract. An algorithm is given for routing the bit-reversal permutation on a shuffle-exchange interconnection network. The algorithm requires 2(log N)-1 stages for N inputs (N is a power of two). The routing scheme is given by an explicit control matrix. Also an adaptation of the bit-reversal algorithm is presented which computes the control bits 'on-line' during the execution of the algorithm.

Keywords and phrases. Perfect shuffle, bit-reversal permutation, interconnection networks, shuffle-exchange, routing algorithm, control matrix.

1. Introduction. The shuffle-exchange interconnection network as proposed by Stone [8] has been shown to be an effective interconnection scheme for parallel computations. Stone [8] presents algorithms for sorting, polynomial evaluation and the FFT, which need log N shuffle-exchange stages on a input of size N ($N = 2^n$). Wu and Feng [10] show that an arbitrary permutation of N inputs can be realized in 3(log N)-1 shuffle-exchange stages, although it is conjectured that 2(log N) stages are always sufficient (Parker [6]).

The bit-reversal permutation is encountered in various algorithms. Implementations of the FFT algorithm of Cooley and Tukey [1] on a perfect shuffle interconnection network, as given by Stone [8] and Cyre and Lipovski [2], produce the resulting Fourier coefficients in reversed binary order. Unscrambling the results is done by the bit-reversal permutation. Wu an Feng [10] prove that an arbitrary permutation of N inputs can be done in 3(log N)-1 shuffle-exchange stages. The algorithm starts with a bit-reversal on the inputs, but it is not explained how this permutation is performed on the shuffle-exchange-network. Wu and Feng [9] also describe the functional relationships among various multi-

stage interconnection networks. The bit-reversal has an important role in simulating other networks on the shuffle-exchange network. For example, the indirect binary n-cube [7] is simulated by a bit-reversal, followed by a n stage shuffle-exchange and a bit-reversal.

Orcutt [5] gives an algorithm that performs the bit-reversal permutation in $O(\sqrt{n})$ steps on a Illiac IV-type interconnection network.

In this paper we present an explicit algorithm for the bit-reversal permutation that needs 2(log N)-1 shuffle-exchange stages. The routing scheme, given by a control matrix, is presented in section 3. In section 4 we show, assuming some processing power in the switches, how the control bits can in fact be computed 'on-line' during the execution of the algorithm. (Parker [6] already proved that 2(log N)-1 shuffle-exchange stages are sufficient for the bit-reversal permutation, but did not provide an explicit control matrix for the algorithm.)

A shuffle-exchange interconnection network of size N consists of multiple copies of a shuffle-exchange stage. At each stage the connection pattern is a perfect shuffle followed by N/2 switches (fig. 1.a). The two inputs of a switch are copied, directly or after exchanging them, to the outputs. In practice a switch will (also) implement a function of its inputs. Whether a switch exchanges its inputs at a certain stage depends on the control bit for that switch at that stage (see fig.1.b).

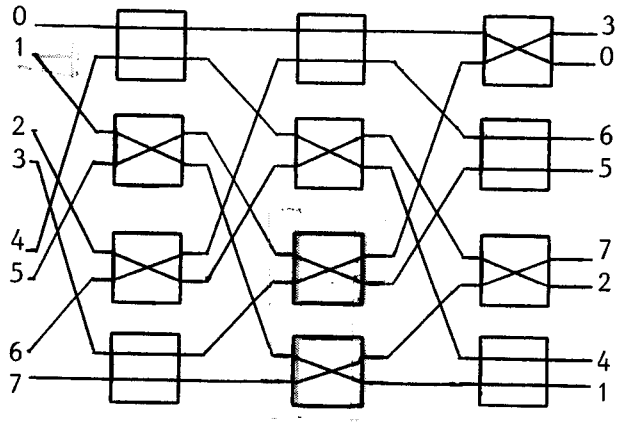

fig.1.a 3-stage shuffle-exchange network of size 8.

$$\begin{bmatrix} 0 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

fig.1.b The control matrix of the network of 1.a.

An algorithm for solving a particular problem on the shuffle-exchange network must describe how the data are routed through the network. A complete description of the routing is given by the control bits of all switches at each stage. The control bits can be given in advance, in a control matrix, or can be computed during the execution of the algorithm. The

algorithm presented in section 2 applies to a multistage shuffle-exchange network. The 'on-line' computation of the control bits, presented in section 4 requires an iterative one stage shuffle-exchange network. That is, a one stage network with the outputs of a switch connected to its inputs (fig.2).

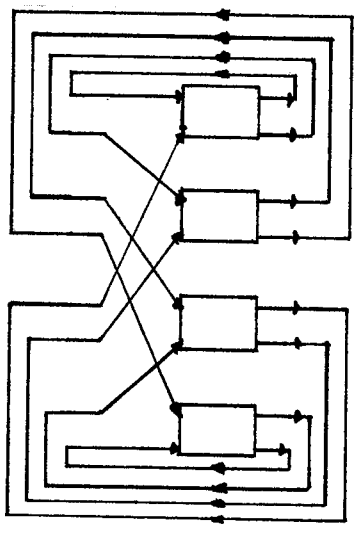

fig.2 An iterative one stage shuffle-exchange network of size 8.

2. The bit-reversal algorithm. In this section we describe the algorithm that computes the bit-reversal permutation for any n-bit integer $x \in \{0,1,\ldots,N-1\}$ $(N = 2^n)$.

Definition. For $x = (x_{n-1}\ x_{n-2} \cdots x_1\ x_0)$, $0 \leq x \leq N-1$, $x \in \mathbb{N}$:

$\rho(x) = (x_0\ x_1 \cdots x_{n-2}\ x_{n-1})$, denotes the bit-reversal permutation,

$\sigma(x) = (x_{n-2} \cdots x_1\ x_0\ x_{n-1})$, denotes the perfect shuffle permutation,

$e(x) = (x_{n-1}\ x_{n-2} \cdots x_1\ \overline{x}_0)$ with $\overline{x} = 0$ if $x = 1$, $\overline{x} = 1$ if $x = 0$,

denotes the exchange.

The bit-reversal permutation on an integer x is described in terms of perfect shuffles on x and exclusive-or operations (XOR) on the bits of x. We distinguish between n odd and n even. The algorithms for each of the cases only differ on minor points. An Algol-like notation is used to describe the algorithms.

Algorithm A1 (n odd).

The input to the network is the set of integers $0,1,\ldots,N-1$ in binary, with integer i sent to the i^{th} input of the shuffle-exchange network $(0 \leq i \leq N-1)$. The outputs are the bit-reversed strings, with $\rho(i)$ at the i^{th} output of the shuffle-exchange network.

begin

1. shuffle $\frac{1}{2}(n-1)$ times;
2. for j from 0 to $\frac{1}{2}(n-1)$ -1
   do shuffle;
      replace $x_{(n-1)/2\ -j}$ by $y_{(n-1)/2\ -j} = x_{(n-1)/2\ -j}$ XOR $x_{(n+1)/2\ +j}$
   od;
3. shuffle;
4. for j from 0 to $\frac{1}{2}(n-1)$ -1
   do shuffle;
      replace x_{n-1-j} by $y_{n-1-j} = x_{n-1-j}$ XOR y_{j+1}
   od;
5. for j from 0 to $\frac{1}{2}(n-1)$ -1
   do shuffle;
      replace $y_{(n-1)/2\ -j}$ by $z_{(n-1)/2\ -j} = y_{(n-1)/2\ -j}$ XOR $y_{(n+1)/2\ +j}$
   od

end.

Performing Algorithm A1 on $x = (x_{n-1}\ x_{n-2} \cdots x_2\ x_1\ x_0)$ we obtain the following forms after each phase, respectively:

1. $(x_{(n-1)/2}\ x_{(n-1)/2\ -1} \cdots x_0\ x_{n-1} \cdots x_{(n+1)/2\ +1}\ x_{(n+1)/2})$
2. $(x_0\ x_{n-1} \cdots x_{(n+1)/2}\ y_{(n-1)/2} \cdots y_2\ y_1)$
3. $(x_{n-1}\ x_{n-2} \cdots y_{(n-1)/2}\ y_{(n-1)/2\ -1} \cdots y_1\ x_0)$
4. $(y_{(n-1)/2}\ y_{(n-1)/2\ -1} \cdots y_1\ x_0 \cdots y_{(n+1)/2\ +1}\ y_{(n+1)/2})$
5. $(x_0\ y_{n-1} \cdots y_{(n+1)/2}\ z_{(n-1)/2} \cdots z_2\ z_1)$

Algorithm A2 (n even).

The same specification as in A1.

begin

1. shuffle $\frac{1}{2}n$ times;
2. for j from 0 to $\frac{1}{2}n-2$
   do shuffle;
      replace $x_{n/2\ -1-j}$ by $y_{n/2\ -1-j} = x_{n/2\ -1-j}$ XOR $x_{n/2\ +1+j}$
   od;
3. shuffle;

4. for j from 0 to $\frac{1}{2}n - 2$
   do shuffle;
      replace x_{n-1-j} by $y_{n-1-j} = x_{n-1-j}$ XOR y_{j+1}
   od;
5. shuffle;
6. for j from 0 to $\frac{1}{2}n - 2$
   do shuffle;
      replace $y_{n/2-1-j}$ by $z_{n/2-1-j} = y_{n/2-1-j}$ XOR $y_{n/2+1+j}$
   od
end.

Performing Algorithm A2 on $x = (x_{n-1} \cdots x_0)$ we obtain:

1. $(x_{n/2-1}\ x_{n/2-2} \cdots x_0\ \ x_{n-1} \cdots x_{n/2+1}\ x_{n/2})$
2. $(x_0\ \ x_{n-1} \cdots x_{n/2}\ \ y_{n/2-1} \cdots y_2\ \ y_1)$
3. $(x_{n-1}\ \ x_{n-2} \cdots y_{n/2-1}\ \ y_{n/2-2} \cdots y_1\ \ x_0)$
4. $(x_{n/2}\ \ x_{n/2-1} \cdots x_0\ \ y_{n-1} \cdots y_{n/2+2}\ y_{n/2+1})$
5. $(y_{n/2-1}\ y_{n/2-2} \cdots y_{n-1}\ \ y_{n-2} \cdots y_{n/2+1}\ x_{n/2})$
6. $(x_0\ \ y_{n-1} \cdots x_{n/2}\ \ z_{n/2-1} \cdots z_2\ \ z_1)$

Theorem 2.1. Algorithm A1 and Algorithm A2 perform the bit-reversal permutation on N inputs in 2(log N)-1 shuffles.

Proof

The y and z were only introduced for notational convenience. For odd as well as even n we defined for all $j = 0, 1, \ldots, \lfloor\frac{n-1}{2}\rfloor - 1$:

$$y_{\lfloor\frac{n-1}{2}\rfloor - j} = x_{\lfloor\frac{n-1}{2}\rfloor - j} \text{ XOR } x_{\lceil\frac{n+1}{2}\rceil + j} \quad (1)$$

and

$$y_{n-1-j} = x_{n-1-j} \text{ XOR } y_{j+1} \quad (2)$$

and, finally,

$$z_{\lfloor\frac{n-1}{2}\rfloor - j} = y_{\lfloor\frac{n-1}{2}\rfloor - j} \text{ XOR } y_{\lceil\frac{n+1}{2}\rceil + j} \quad (3)$$

If we define $k = \lfloor \frac{n-1}{2} \rfloor - 1 - j$ then for all $k = 0,1, \ldots, \lfloor \frac{n-1}{2} \rfloor - 1$ (1) can be written as

(1') $y_{k+1} = x_{k+1}$ XOR x_{n-1-k}

Using this for rewriting (2) and (3) we obtain

(2') $y_{n-1-k} = x_{n-1-k}$ XOR y_{k+1}

$= x_{n-1-k}$ XOR $(x_{k+1}$ XOR $x_{n-1-k})$

$= x_{k+1}$

and,

(3') $z_{k+1} = y_{k+1}$ XOR y_{n-1-k}

$= y_{k+1}$ XOR x_{k+1}

$= (x_{k+1}$ XOR $x_{n-1-k})$ XOR x_{k+1}

$= x_{n-1-k}$

Substitution of the y's and z's by the corresponding x values in the results after applying Algorithms A1 and A2 to $x = (x_{n-1} \ldots x_0)$ shows that both algorithms compute the bit-reversal $\rho(x) = (x_0\ x_1 \ldots x_{n-1})$. The number of shuffles performed by Algorithm A1 is 2n-1, since each of phases 1, 2, 4 and 5 takes $\frac{1}{2}(n-1)$ shuffles and phase 3 needs one more shuffle. By simular argument algorithm A2 takes 2n-1 shuffles as well. □

In the next section we show how the algorithms can be implemented on a shuffle-exchange network.

3. The control matrix for the bit-reversal permutation on a shuffle-exchange network. A control matrix describes at any stage of the network the behaviour of the switches $(P_0, P_1, \ldots, P_{\frac{1}{2}N-1})$. All switches at a stage must be set at the same time. Every switch has two inputs. If the switch is set to 0 it simply copies its inputs to its outputs. A switch that is set to 1 exchanges its inputs before they are output. A "conflict" occurs if for one input the switch must be 1 and for the other it must be 0. In assigning values to the switches in a certain stage we have to avoid such conflicts.

Definition. $C^{odd} = [C_1^{odd}\ C_2^{odd}\ C_3^{odd}\ C_4^{odd}\ C_5^{odd}]$

$C^{even} = [C_1^{even}\ C_2^{even}\ C_3^{even}\ C_4^{even}\ C_5^{even}\ C_6^{even}]$

both binary $\frac{1}{2}N \times (2n-1)$ matrices ($N = 2^n$). C_i^{odd} and C_i^{even} are the control matrices for phase i of Algorithm A1 and Algorithm A2, respectively, ($i = 1, 2, \ldots, 5, 6$).

The j^{th} columns of C_i^{odd} and C_i^{even} ($i = 1, 2, \ldots, 5, 6$) have the values of the switches after the $(j+1)^{st}$ shuffle in phase i ($j = 0, 1, \ldots, \lfloor\frac{n-1}{2}\rfloor - 1$).

Definition. θ is the column vector of length $\frac{1}{2}N$ with all entries zero.

Z is the $\frac{1}{2}N \times \lceil\frac{n-1}{2}\rceil$ matrix with all entries zero.

Lemma 3.1. $C_1^{even} = Z$ and $C_1^{odd} = Z$.

Proof.

Trivial, from the description of the algorithms. □

Lemma 3.2. $C_3^{even} = C_5^{even} = \theta$ and $C_3^{odd} = \theta$.

Proof.

Trivial, from the description of the algorithms. □

Definition. ψ is a binary $\frac{1}{2}N \times (n-1)$ matrix with r^{th} row equal to the binary representation of r ($r = 0, \ldots, \frac{1}{2}N-1$). Also: ψ_i is the $(n-2-i)^{th}$ column of ψ ($i = 0, \ldots, n-2$).

$\psi^{(1)} = [[\psi_{2j+1}]\ j = 0, \ldots, \lfloor\frac{n-1}{2}\rfloor - 1]$

$\psi^{(2)} = [[\psi_{2j}]\ \ j = 0, \ldots, \lfloor\frac{n-1}{2}\rfloor - 1]$

Example. $N = 16$, $n = 4$, $\psi = [\psi_2\ \psi_1\ \psi_0] = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$,

$\psi^{(1)} = [\psi_1]$, $\psi^{(2)} = [\psi_0]$.

Note that the value of the bit in position $x = (x_{n-2}\ x_{n-3} \cdots x_0)$ of column vector ψ_i is x_i ($i = 0, 1, \ldots, n-2$).

Lemma 3.3. $C_2^{odd} = C_5^{odd} = \psi^{(2)}$.

Proof.

Assume we started Algorithm A1 with $x = (x_{n-1} \cdots x_0)$. After the $\frac{1}{2}(n-1)$ shuffles of phase 1 and i (i=0,1, ..., $\frac{1}{2}(n-1)-1$) executions of the 'shuffle-replace'-loop the $i+1^{st}$ execution of the loop is started with a shuffle (note that the loopcounter j=i). In the resulting sequence:

$(x_{\frac{n-1}{2}-(j+1)} \cdots x_0\ x_{n-1} \cdots x_{\frac{n+1}{2}+j} \cdots x_{\frac{n+1}{2}}\ y_{\frac{n-1}{2}} \cdots y_{\frac{n-1}{2}-(j-1)}\ x_{\frac{n-1}{2}-j})$

the rightmost bit with value $x_{\frac{n-1}{2}-j}$ should be replaced by $y_{\frac{n-1}{2}-j} = x_{\frac{n-1}{2}-j}$ XOR $x_{\frac{n+1}{2}+j}$, i.e., if $x_{\frac{n+1}{2}+j} = 1$ then $x_{\frac{n-1}{2}-j}$ is replaced by $\overline{x}_{\frac{n-1}{2}-j}$ otherwise no replacement is needed.

In the following bits in a bitsequence will be numbered starting with the rightmost bit at 0. Whether a replacement of the rightmost bit is needed only depends on bit number 2j+1 in the sequence. The same observation holds for the replacement of the rightmost bit after the shuffle in the $i+1^{st}$ execution of the loop in phase 5 of Algorithm A1.

Divide the N bitstrings of length n in disjunct subsets of size 2^{2j+2}. Each subset is divided in two also disjunct parts (fig. 3):

$$S_k = \{(x_{n-1} \cdots x_{2j+2}\ x_{2j+1} \cdots x_0) \mid (x_{n-1} \cdots x_{2j+2}) = k\},$$
for $k = 0, 1, \ldots, 2^{n-2j-2} - 1$.

$$S_k^{(0)} = \{(x_{n-1} \cdots x_{2j+2}\ x_{2j+1} \cdots x_0) \in S_k \mid x_{2j+1} = 0\}.$$

$$S_k^{(1)} = \{(x_{n-1} \cdots x_{2j+2}\ x_{2j+1} \cdots x_0) \in S_k \mid x_{2j+1} = 1\}.$$

The only pairs of sequences that can be exchanged in a certain stage of the shuffle-exchange network are those that are equal in all but the rightmost bit. Both sequences in such a pair belong to the same S_k and both are in $S_k^{(0)}$ or both are in $S_k^{(1)}$. This follows from the even number of sequences in S_k, $S_k^{(0)}$ and $S_k^{(1)}$.

$$
S_k \left\{ \begin{array}{l}
S_k^{(0)} \left\{ \begin{array}{llllllll}
x_{n-1} & x_{n-2} & \cdots & x_{2j+2} & 0 & 0 & \cdots & 0\ 0 \\
x_{n-1} & x_{n-2} & \cdots & x_{2j+2} & 0 & 0 & \cdots & 0\ 1 \\
x_{n-1} & x_{n-2} & \cdots & x_{2j+2} & 0 & 0 & \cdots & 1\ 0 \\
x_{n-1} & x_{n-2} & \cdots & x_{2j+2} & 0 & 0 & \cdots & 1\ 1 \\
\vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\
x_{n-1} & x_{n-2} & \cdots & x_{2j+2} & 0 & 1 & \cdots & 1\ 1
\end{array} \right\} 2^{2j+1} \text{ bitstrings} \\
S_k^{(1)} \left\{ \begin{array}{llllllll}
x_{n-1} & x_{n-2} & \cdots & x_{2j+2} & 1 & 0 & \cdots & 0\ 0 \\
x_{n-1} & x_{n-2} & \cdots & x_{2j+2} & 1 & 0 & \cdots & 0\ 1 \\
x_{n-1} & x_{n-2} & \cdots & x_{2j+2} & 1 & 0 & \cdots & 1\ 0 \\
x_{n-1} & x_{n-2} & \cdots & x_{2j+2} & 1 & 0 & \cdots & 1\ 1 \\
\vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\
x_{n-1} & x_{n-2} & \cdots & x_{2j+2} & 1 & 1 & \cdots & 1\ 1
\end{array} \right\} 2^{2j+1} \text{ bitstrings}
\end{array} \right.
$$

fig. 3. Partition of the set of bitstrings of length n.

For Algorithm A1 we have to exchange pairs of sequences with bit number 2j+1 equal to 1. The set of sequences with this property is $\bigcup_{k=0}^{2^{n-2j-2}} S_k^{(1)}$. No conflicts will occur because only pairs from this set will be exchanged. The sequences of $S_k^{(1)}$ are input to the switches $P_{(2k+1)2^{2j}} \cdots P_{(2k+2)2^{2j}-1}$. The corresponding entries in the j^{th} column of the control matrix are 1, the other entries $P_{(2k)2^{2j}} \cdots P_{(2k+1)2^{2j}-1}$ are 0 ($k = 0, \ldots, 2^{n-2j-2}$). The resulting column vector is ψ_{2j} ($j = 0, 1, \ldots, \lfloor \frac{n-1}{2} \rfloor - 1$). Thus, $C_2^{odd} = \psi^{(2)}$, and $C_5^{odd} = \psi^{(2)}$. □

Lemma 3.4. $C_4^{odd} = \psi^{(1)}$, $C_2^{even} = C_4^{even} = C_6^{even} = \psi^{(1)}$

Proof.

In the $(i+1)^{st}$ execution of the loop of phase 4 of Algorithm A1, and phases 2, 4 and 6 of Algorithm A2 the decision on replacing the rightmost bit of the current sequence is made according to the current value of bit number 2j+2. Define

$T_l = \{(x_{n-1}\cdots x_{2j+3}\ x_{2j+2}\cdots x_0) \mid (x_{n-1}\cdots x_{2j+3}) = l\}$, $l = 0,\ldots,2^{n-2j-3}-1$,

$T_l^{(0)} = \{(x_{n-1}\cdots x_{2j+3}\ x_{2j+2}\cdots x_0) \in T_l \mid x_{2j+2} = 0\}$,

$T_l^{(1)} = \{(x_{n-1}\cdots x_{2j+3}\ x_{2j+2}\cdots x_0) \in T_l \mid x_{2j+2} = 1\}$.

Now we divide the N bitstrings of length n in 2^{n-2j-3} disjunct subsets of size 2^{2j+3}, and we split these T_l as we did before in the proof of Lemma 3.3.

All sequences in which the rightmost bit must be replaced are in $\bigcup_{l=1}^{2^{n-2j-3}} T_l^{(1)}$. These sequences are input to the switches $P_{(2l+1)2^{2j+1}}, \ldots, P_{(2l+2)2^{2j+1}-1}$. The corresponding entries in the j^{th} column of C_4^{odd}, C_2^{even}, C_4^{even} and C_6^{even} are 1, the other entries are 0. This results in a column vector ψ_{2j+1} $(j = 0, 1, \ldots, \lfloor\frac{n-1}{2}\rfloor - 1)$, and the control matrix $\psi^{(1)}$ for each of the phases mentioned. □

Theorem 3.5. The control matrix for the bit-reversal permutation on the shuffle-exchange network with 2^n inputs is given by:

$C^{odd} = [[Z]\ [\psi^{(2)}]\ [\theta]\ [\psi^{(1)}]\ [\psi^{(2)}]]$ for n odd,

$C^{even} = [[Z]\ [\psi^{(1)}]\ [\theta]\ [\psi^{(1)}]\ [\theta]\ [\psi^{(1)}]]$ for n even.

Proof.

From Lemma 3.1., 3.2., 3.3., and 3.4. □

Examples. $n = 3$, $N = 8$, $C = \begin{bmatrix} 0&0&0&0&0 \\ 0&1&0&0&1 \\ 0&0&0&1&0 \\ 0&1&0&1&1 \end{bmatrix}$

$n = 4$, $N = 16$, $C = \begin{bmatrix} 0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0 \\ 0&0&1&0&1&0&1 \\ 0&0&1&0&1&0&1 \\ 0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0 \\ 0&0&1&0&1&0&1 \\ 0&0&1&0&1&0&1 \end{bmatrix}$

4. Runtime computation of the controlbits for the switches. In this section we assume we have an iterative one stage shuffle-exchange network in which the switches have some processing power and some registers. The processors in the switches execute in parallel. Each of the processors has seven registers. ID is a n bit register which contains the number of the processor. Bit number i in register ID is denoted by ID[i]. Four registers of size n are used for input and output (IN1, IN2, OUT1, OUT2). The value n is kept in a register LOGN. LOGN and ID are read-only registers. One more register CNT is used for counting.

Algorithm A1* (n odd).

```
begin
        CNT := 0;
label 1: (*phase 1*)
        read inputs;
        OUT1 := IN1 ; OUT2 := IN2;          (* copy input to output *)
        write outputs;
        CNT := CNT + 2;
        if CNT < LOGN -1 then goto label 1 fi;
        CNT := 0;
label 2: (*phase 2*)
        read inputs;
        if ID[CNT]  = 0 then OUT1 := IN1 ; OUT2 := IN2  (* copy *)
                        else OUT1 := IN2 ; OUT2 := IN1  (* exchange *)
        fi;
        write outputs;
        CNT := CNT + 2;
        if CNT < LOGN -1 then goto label 2 fi;
label 3: (*phase 3*)
        read inputs;
        OUT1 := IN1 ; OUT2 := IN2;                      (* copy *)
        write outputs;
        CNT := 1;
```

```
label 4: (*phase 4*)
         read inputs;
         if ID[CNT]  = 0 then OUT1 := IN1 ; OUT2 := IN2     (* copy *)
                         else OUT1 := IN2 ; OUT2 := IN1     (* exchange *)
         fi;
         write outputs;
         CNT := CNT + 2;
         if CNT < LOGN -1 then goto label 4 fi;
         CNT := 0;
label 5: (*phase 5*)
         read inputs;
         if ID[CNT]  = 0 then  OUT1 := IN1 ; OUT2 := IN2    (* copy *)
                         else  OUT1 := IN2 ; OUT2 := IN1    (* exchange *)
         fi;
         write outputs;
         CNT := CNT + 2;
         if CNT < LOGN -1 then goto label 5 fi
end.
```

Lemma 4.1. Algorithm A1* computes the bit-reversal on $N = 2^n$ (n odd) inputs in 2(log N)-1 shuffles. The control bits are computed at each stage of the shuffle-exchange network in O(1) time.

Proof.

In phases 2, 4 and 5 the decision on exchanging the inputs is taken according to the value of a bit in register ID of the switch. The number of the bit in ID to inspect is in the register CNT. This is the implementation of the note in section 3 after the definition of ψ (p. 7). The rest of the algorithm is identical to Algorithm. A1. □

For even n Algorithm A2 can be adapted similarly. Finally we note that Lang and Stone [3] introduced a shuffle-exchange network with simplified control. In this network switches perform a binary operation on the two control bits that enter the switch with the inputs. The processing power required by Algorithm A1* is more complex. We need integer addition, selection of a particular bit in a register, and comparisons. Also the different phases of the algorithm execute different statements.

5. Conclusion. We presented a routing algorithm that produces the bit-reversal permutation of N inputs in 2(log N)-1 shuffle-exchange stages. The control bits of the switches are given in a control matrix or can be computed during the execution of the algorithm. Consequently, the FFT algorithm, including the unscrambling of the results, can be computed in 3(log N)-1 stages on the shuffle-exchange network, and the indirect binary n-cube can be simulated in 5(log N)-1 shuffle-exchange stages.

References.

[1] Cooley, J.W. and J.W. Tukey, An algorithm for machine calculation of complex Fourier series, Math. Comput., vol.19, pp. 297-301, April 1965.

[2] Cyre, W.R. and G.J. Lipovski, On generating multipliers for a cellular fast Fourier transform processor, IEEE Tr. on Comp., vol.C-21, pp. 83-87, Jan. 1972.

[3] Lang, T. and H.S. Stone, A shuffle-exchange network with simplified control, IEEE Tr. on Comp., vol.C-25, pp. 55-65, Jan. 1976.

[4] Lenfant, J., Parallel permutations of data: A Benes network control algorithm for frequently used permutations, IEEE Tr. on Comp., vol.C-27, pp. 637-647, July 1978.

[5] Orcutt, S.E., Implementation of Permutation Functions in Illiac IV-Type Computers, IEEE Tr. on Comp., Vol.C-25, pp.929-936, Sept.1976.

[6] Parker, S.D. jr., Notes on the shuffle-exchange-type switching networks, IEEE Tr. on Comp., vol.C-29, March 1980.

[7] Pease, M.C., The indirect binary n-cube micro processor array, IEEE Tr. on Comp., vol.C-26, pp. 548-573, May 1977.

[8] Stone, H.S., Parallel processing with the perfect shuffle, IEEE Tr. on Comp., vol.C-20, pp. 153-161, Feb. 1971.

[9] Wu, C-L. and T-Y. Feng, The reverse-exchange interconnection network, IEEE Tr. on Comp., vol.C-29, pp. 801-811, Sept. 1980.

[10] Wu, C-L. and T-Y. Feng, The universality of the shuffle-exchange network, IEEE Tr. on Comp., vol.C-30, pp. 324-332, May 1981.