

Philips Research Laboratory
Av. Van Becelaere 2, Box 8
B- 1170 Brussels - Belgium

TECHNICAL NOTE N141

Multiconnection networks

C. RONSE
December 1980

ABSTRACT

A multiconnection network is a switching network which can connect each of its outputs to some input. We investigate designs able to realize any connection of the outputs to the inputs. The complexity bounds are $O(N \log N)$ for the cost and $O(\log N)$ for the delay, but in practice, a network with an easy architecture and algorithm requires a cost and a delay of the form $O(N^2)$ and $O(N)$ respectively.

MULTICONNECTION NETWORKS

I. Introduction

Let M and N be two positive integers. Consider a switching network S with M data inputs I_0, \dots, I_{M-1} , N data outputs O_0, \dots, O_{N-1} , and several control inputs which can realize input-output behaviors of the following type :

There is a map $f : Z_N \rightarrow Z_M$ such that for any $x \in Z_N$ and $y \in Z_M$, I_y is connected to O_x if and only if $y = f(x)$. (1)

(Here $Z_k = \{0, \dots, k-1\}$ for any integer $k > 0$).

We call such a connection a multiconnection, because an input can be connected to several outputs. We say also that S is a multiconnection network. The multiconnection (1) can be written $[f]$ and we say then that S realizes $[f]$ (or simply : S realizes f).

The number of multiconnections realized by S is called the multiplicity of S . If S can realize all multiconnections, then S is total. Its multiplicity is then equal to :

$$M^N \quad (2)$$

Finally, we say that S is square if $N=M$.

Note. We use the same notation as in [2].

II. Design methods

A. The direct design

Suppose first that S has multiplicity k , where $k \leq M$. Then S can be built with N multiplexers on k bits having the same control inputs.

Indeed take N such multiplexers M_0, \dots, M_{N-1} . Suppose that S realizes the multiconnections $[f_0], \dots, [f_{k-1}]$. Then we make the following connections :

- For $i=0, \dots, N-1$, the output of M_i is connected to O_i .
- For $j=0, \dots, k-1$ and $i=0, \dots, N-1$, the data input j of M_i is connected to $I_{f_j}(i)$.

Then if the control inputs of M_0, \dots, M_{N-1} are set in the state j (where $j=0, \dots, k-1$), then each M_i ($i=0, \dots, n-1$) connects its output to its data input j and so O_i is connected to $I_{f_j}(i)$. Thus S realizes f_j .

Suppose now that the multiplicity k of S is larger than M . Then S can be built with N multiplexers on M bits having pairwise distinct control inputs. We take such N multiplexers M_0, \dots, M_{N-1} , and for $i=0, \dots, N-1$ and $j=0, \dots, M-1$, we connect :

- The output of M_i with O_i
- The data input j of M_i to I_j .

Then the resulting multiconnection network is total. Indeed, for any map $f : Z_N \rightarrow Z_M$, if we set each M_i in the state $f(i)$ ($i=0, \dots, N-1$), then O_i is connected to $I_{f(i)}$ and S realizes f .

These two methods are illustrated on Figures 1 and 2 for $N=4$, $M=5$ and $k=3$ in the first case.

B. The three stage design

This design for total multiconnection networks was found in [4]. Although only square networks were considered, it is also valid for $N < M$.

It consists of three stages :

- (1) A permutation network on M bits.
- (2) A branching network on N bits.
- (3) A permutation network on N bits.

A branching network on N bits is a particular square multiconnection network on N bits which can realize every multiconnection [f] such that :

There are k positive numbers a_0, \dots, a_{k-1} such that :

$$0 = a_0 < \dots < a_{k-1} \leq N$$

and $f(x) = \max\{a_i \mid (0 \leq i \leq k-1) \mid a_i \leq x\}$. (3)

Such a multiconnection f is called a branching. An example of branching is given in Figure 3 for $N=7$. The three-stage network is illustrated in Figure 4 for $N=4$ and $M=5$.

We will now describe briefly the control algorithm of this design.

A map $f : Z_N \rightarrow Z_M$ can be described by a sequence of N elements of Z_M , which are the images by f of $0, 1, \dots, N-1$. Now f can also be described in the inverted form, that is by a sequence of M subsets of Z_N , which are the inverse images by f of $0, 1, \dots, M-1$. Any of these two forms can be obtained from the other by scanning through the sequence. This operation has cost equal to N.

Suppose now that f is given in the inverted form. There are k numbers y_0, \dots, y_{k-1} such that $f^{-1}(y_i) = \{x \in Z_N \mid f(x) = y_i\}$ is nonvoid.

Write :

$$f^{-1}(y_i) = \{x_0^i, \dots, x_{b_i^i-1}^i\} \quad (4)$$

for $i=0, \dots, k-1$.

Then set :

$$a_0 = 0 \quad (5)$$

and for $i=1, \dots, k-1$

$$a_i = \sum_{j=0}^{i-1} b_j \quad (6)$$

We make then the following operations on the three stages :

- (1°) In the first stage, we connect for $i=0, \dots, k-1$ the input y_i to the output a_i and complete the connection in order to get a permutation.
- (2°) In the second stage, we operate the branching (3) determined by the values of a_0, \dots, a_{k-1} chosen in (5) and (6)
- (3°) In the third stage, we connect for every $i=0, \dots, k-1$, the input $a_i + u$ ($0 \leq u < b_i$) to the output x_u^i .

Then the network realizes f .

An example of this decomposition is given in Figure 5 for $N=4$ and $M=5$.

For $M < N$, the network can be designed by adding $N-M$ iddle inputs, so that we get $M \geq N$.

Now we have only to describe possible designs for a branching network on N bits.

Figure 6 gives the design of a branching network on 2 bits, also called a branching cell, using one multiplexer. It also gives its two states : "through" and "branching".

In [4], the authors give a design for a branching network on N bits for $N > 2$. It is illustrated in Figure 7 for $N=6$.

This design consists in $N-1$ branching cells placed in $N-1$ stages. If we label these cells $1, \dots, N-1$ (as in Figure 7), then the branching (3) can be realized by setting the cells a_1, \dots, a_{k-1} in the "through" state, and the other cells in the "branching" state. In other words, we set a cell i in the "through" state if $f(i-1) < f(i)$ and in the "branching" state if $f(i-1) = f(i)$, where $[f]$ is the branching that we want to realize.

There is also another design having cost around $N \log_2 N$ and delay $2 \log_2 N$ in terms of branching cells. We will describe it here :

For any number u , let $\lceil u \rceil$ be the smallest integer larger than or equal to u . Let $n = \lceil \log_2 N \rceil$. Then we get a branching network by connecting successively the following $2n-1$ stages :

Stage $2k$ ($k=0, \dots, n-2$) : For every $i \in \{0, \dots, N-1-2^k\}$ such that $i \equiv 2^k, \dots, 2^{k+1}-1 \pmod{2^{k+1}}$, we connect the levels i and $i+2^k$ by a branching cell.

Stage $2k+1$ ($k=0, \dots, n-2$) : For every $i \in \{0, \dots, N-1-2^k\}$ such that $i \equiv 0, \dots, 2^k-1 \pmod{2^{k+1}}$, we connect the levels i and $i+2^k$ by a branching cell.

Stage $2(n-1)$: For every $i \in \{0, \dots, N-1-2^{n-1}\}$, we connect the levels i and $i+2^{n-1}$ by a branching cell.

This construction is illustrated in Figure 8 for $N=10$.

The control algorithm is easy. Given a branching cell between the levels i and j (where $i < j$), we set it in the following states :

- If $f(i) = f(j)$, the "branching state.
- If $f(i) < f(j)$, the "through" state.

(Here $[f]$ is the branching that we want to realize, see (3)).

Let us explain briefly why this design works. The choice of the congruences in the stages $2k$ and $2k+1$ for $k < n-1$ ensures that every level is connected to at most one branching cell by stage.

Suppose that $f(j) = i$ for $j = i + \sum_{r=0}^{n-1} d_r 2^r$. Then the stages 0 and 1, 2 and 3, ..., $2(n-2)$ and $2(n-2)+1$, and finally $2(n-1)$, will connect together the levels i and $i+d_0$, $i+d_0$ and $i+d_0+2d_1, \dots, i+d_0+\dots+d_{n-3}2^{n-3}$ and $i+d_0+\dots+d_{n-3}2^{n-3}+d_{n-2}2^{n-2}$ and finally $i+d_0+\dots+d_{n-3}2^{n-2}$ and $i+d_0+\dots+d_{n-1}2^{n-1} = j$. Thus the output j will be connected to the input i by a succession of branchings.

This network has $2n-1$ stages. Let us now count the number of cells it contains.

For every $k=0, \dots, n-1$, there is a branching cell between the stages i and $i+2^k$ if and only if :

$$i \in \{0, \dots, N-1-2^k\}. \quad (7)$$

Thus the number of cells is

$$\begin{aligned} \sum_{k=0}^{n-1} (N-2^k) &= nN - \sum_{k=0}^{n-1} 2^k \\ &= nN - (2^n - 1) \\ &= nN - 2^n + 1. \end{aligned} \quad (8)$$

This number is asymptotically equal to $N \log_2 N$.

Note that for $N=2$, a total multiconnection network can be built with only two stages : first a permutation cell and then a branching cell (see Figure 9).

C. The use of Clos and related networks

Let A and B two switching networks on a and b bits respectively. Then we can construct the Clos Network $B \times A$ by taking three stages connected by perfect shuffles, the first one consisting of a copies of B, the second one of b copies of A, the last one of a copies of B (see Figure 10).

If A and B are multiconnection networks, then $B \times A$ is also a multiconnection network. But when is $B \times A$ total ? We have the following result :

Theorem 1. Let A and B be total multiconnection networks on a and b bits respectively ($a, b \geq 2$). Then $B \times A$ is total if and only if $a=2$.

To prove it, we will consider three cases :

(1°) $3 \leq a \leq b$.

(2°) $a > b$.

(3°) $a = 2$.

We will write B_0, \dots, B_{a-1} for the a copies of B in the first stage of $B \times A$, A_0, \dots, A_{b-1} for the b copies of A in the second one, and B_0^1, \dots, B_{a-1}^1 for the a copies of B in the third one.

Let us take the first case : $3 \leq a \leq b$. We can write $b = a+c$. Suppose that we want to realize the function f defined in the following way :

For any $x \in Z_a$ and $y \in Z_b$,

$$f(xb+y) = xb+y \quad \text{if} \quad y \notin \{c, c+1, c+2\}. \quad (9)$$

$$= c \quad \text{if} \quad y=c. \quad (10)$$

$$= b+c+x \quad \text{if} \quad y=c+1. \quad (11)$$

$$= c+x+1 \quad \text{if} \quad y=c+2. \quad (12)$$

It is easily checked that for any $x \in Z_a$ and $y, y^1 \in Z_b$, $f(xb+y) \neq f(xb+y^1)$ whenever $y \neq y^1$. Thus B_0^1, \dots, B_{a-1}^1 are in the permutation mode.

As $0, \dots, 2b-1$ are all in the image of f , B_0 and B_1 must be in the permutation mode.

Thus, the input c of B_0 is connected to only one output of B_0 , say x . Thus it is connected to the input 0 of A_x . Now the output c of each B_1^i is connected to only one input of B_1^i , which must be connected to the input 0 of A_x . Thus it is the input x of B_1^i and so A_x is in the branching mode : every output is connected to the input 0 .

As B_1 is in the permutation mode, its output x must be connected to one of its inputs, say y , and it is the only output connected to this input. But then no path can go from this input to the third stage, because no output of A_x is connected to its input 1 . But this contradicts the fact that $b+y \in \text{Im}f$.

This case is illustrated in Figure 11.

Let us take now the second case : $a > b$. Then we define f in the following way :

For any $x \in Z_a$ and $y \in Z_b$,

$$f(xb+y) = 0 \quad \text{if} \quad y=0. \quad (13)$$

$$= b+x \quad \text{if} \quad x < b \text{ and } y=1. \quad (14)$$

$$= y \quad \text{if} \quad x = b+1. \quad (15)$$

$$= xb+y \quad \text{otherwise.} \quad (16)$$

Then we apply the same argument as in the first case; but where we replace c by 0 .

Let us now consider the third case : $a=2$. We will show that $B \times A$ is total. We first prove the following result :

Lemma 2. Let n be a positive integer. Let X_0, X_1, Y_0 and Y_1 be sets such that $X_0 \cap X_1 = Y_0 \cap Y_1 = \emptyset$ and $|Y_0| = |Y_1| = n \geq \max\{|X_0|, |X_1|\}$. Let f be a map $X_0 \cup X_1 \rightarrow Y_0 \cup Y_1$. Then either :

a) $\text{Im}f \cap Y_1 = \emptyset,$

b) $\text{Im}f \cap Y_0 = \emptyset,$

or c) there exist $y_0 \in Y_0, y_1 \in Y_1, Z_0 \subseteq X_0$ and $Z_1 \subseteq X_1$ such that :

(i) $\{f(Z_0), f(Z_1)\} \subseteq \{\emptyset, \{y_0\}, \{y_1\}\}.$

(ii) $n-1 \geq \max\{|X_0| - |Z_0|, |X_1| - |Z_1|\}.$

(iii) If g is the restriction of f to $(X_0 \setminus Z_0) \cup (X_1 \setminus Z_1)$, then

$$n-1 \geq \max\{|Y_0 \cap \text{Im}g|, |Y_1 \cap \text{Im}g|\}.$$

Proof. Suppose that (a) and (b) do not hold. We have the following five cases :

(1°) For some $j=0$ or 1 , there exist $y \in Y_j$ and $y^1 \in Y_{1-j}$ such that $f^{-1}(y) \cap X_0 \neq \emptyset \neq f^{-1}(y) \cap X_1$ and $f^{-1}(y^1) = \emptyset$.

Then we take $Z_i = f^{-1}(y) \cap X_i$ ($i=0,1$) and so (c) holds with $y = y_j$ and $y^1 = y_{1-j}$ because $y, y^1 \notin \text{Im}g$.

(2°) For some $j=0$ or 1 , there exist $y, z \in Y_j$ and $y^1 \in Y_{1-j}$ such that $f^{-1}(z) = \emptyset, f^{-1}(y) \cap X_0 \neq \emptyset \neq f^{-1}(y) \cap X_1$ and $\emptyset \neq f^{-1}(y^1) \subseteq X_i$ for some $i=0$ or 1 .

Then we take $Z_i = f^{-1}(y^1)$ and $Z_{1-i} = f^{-1}(y) \cap X_{1-i}$ and so (c) holds with $y=y_j$ and $y^1=y_{1-j}$, since $z, y^1 \notin \text{Im}g$.

(3°) There exist $y \in Y_0$ and $y^1 \in Y_1$ such that $\emptyset \neq f^{-1}(y) \in X_j$ and $\emptyset \neq f^{-1}(y^1) \in X_{1-j}$ for some $j=0$ or 1 .

Then we take $Z_j = f^{-1}(y)$ and $Z_{1-j} = f^{-1}(y^1)$ and so (c) holds with $y = y_0$ and $y^1 = y_1$, because $y, y^1 \notin \text{Img}$.

(4°) For some $i, j \in \{0, 1\}$, $X_i = \emptyset$ and there exist $y \in Y_j$ and $y^1 \in Y_{1-j}$ such that $f^{-1}(y) = \emptyset$ and $f^{-1}(y^1) \neq \emptyset$.

Then we take $Z_i = \emptyset$ and $Z_{1-i} = f^{-1}(y^1) (\subseteq X_{1-i} \text{ since } X_i = \emptyset)$. Then (c) holds with $y=y_j$ and $y^1=y_{1-j}$, since $n-1 \geq |X_i| = |X_i| - |Z_i|$ and $y, y^1 \notin \text{Img}$.

(5°) None of the preceding cases holds. Then we show that we have a contradiction. As (3°) does not hold, f may not be a bijection and so for some $j=0$ or 1 , there is some $z \in Y_j \setminus \text{Im}f$. As (a) and (b) do not hold, there exist some $y \in Y_j \cap \text{Im}f$ and $y^1 \in Y_{1-j} \cap \text{Im}f$. As (1°) does not hold, $\emptyset \neq f^{-1}(y^1) \subseteq X_0$ or X_1 . Now, as (2°) does not hold, $\emptyset \neq f^{-1}(y) \subseteq X_0$ or X_1 . But (3°) does not hold. Thus $f^{-1}(y) \cup f^{-1}(y^1) \subseteq X_i$ for some $i=0$ or 1 . Now this argument can be repeated if we replace y by any $\tilde{y} \in Y_j \cap \text{Im}f$ or y^1 by any $\tilde{y}^1 \in Y_{1-j} \cap \text{Im}f$. Thus $f^{-1}(Y_j) \cup f^{-1}(Y_{1-j}) \subseteq X_i$ and so $X_{1-i} = \emptyset$, and so (4°) holds, which is a contradiction.

Now the following result will allow us to prove Theorem 1.

Proposition 3. Let b be a positive integer. Let X_0, X_1, Y_0 and Y_1 be sets such that $X_0 \cap X_1 = \emptyset = Y_0 \cap Y_1$ and $|Y_0| = |Y_1| = b \geq \max\{|X_0|, |X_1|\}$. Let f be a map $X_0 \cup X_1 \rightarrow Y_0 \cup Y_1$. Then there exists $y_{0,0}, \dots, y_{0,b-1} \in Y_0$ and $y_{1,0}, \dots, y_{1,b-1} \in Y_1$ (in each case not necessarily pairwise distinct), $Z_{0,0}, \dots, Z_{0,b-1} \subseteq X_0$ and $Z_{1,0}, \dots, Z_{1,b-1} \subseteq X_1$ such that :

- (i) $X_0 = Z_{0,0} \cup \dots \cup Z_{0,b-1}$.
- (ii) $X_1 = Z_{1,0} \cup \dots \cup Z_{1,b-1}$.
- (iii) For every $i=0,1,\dots,b-1$, $\{f(Z_{0,i}), f(Z_{1,i})\} \subseteq \{\emptyset, \{y_{0,i}\}, \{y_{1,i}\}\}$.

Proof. We use induction b . The result is obvious for $b=1$. Suppose now that $b > 1$ and that the result is true for $b-1$. Then we can apply Lemma 2 with $n=b$.

If (a) holds, then write $y_{j,i}$ for the i th element of Y_j ($j=0,1$; $i=0,\dots,b-1$) and take $Z_{j,i} = f^{-1}(y_{0,i}) \cap X_j$. Then the result holds.

A similar argument works if (b) holds. Suppose now that (c) holds. Set $Z_j = Z_{j,b-1}$ and $y_j = y_{j,b-1}$ ($j=0,1$). Choose some $Y_0^1 \subseteq Y_0$ and $Y_1^1 \subseteq Y_1$ such that $|Y_0^1| = |Y_1^1| = n-1$, $Y_0^1 \supseteq Y_0 \cap \text{Img}$ and $Y_1^1 \supseteq Y_1 \cap \text{Img}$. If we write $X_0^1 = X_0 \setminus Z_0$ and $X_1^1 = X_1 \setminus Z_1$, then $X_0^1, X_1^1, Y_0^1, Y_1^1$ and g satisfy the hypothesis of the Proposition, but with $b-1$ instead of b . Then we can find $y_{0,0}, \dots, y_{0,b-2}, y_{1,0}, \dots, y_{1,b-2}, z_{0,0}, \dots, z_{0,b-2}, z_{1,0}, \dots, z_{1,b-2}$ satisfying (i), (ii) and (iii). As Z_0, Z_1, Y_0 and Y_1 satisfy result (c) of Lemma 2, and as g is the restriction of f to $X_0^1 \cup X_1^1$, (i), (ii) and (iii) hold also for X_0, X_1 and f .

In fact, we can derive from Proposition 3 the fact that if $X_0, X_1, Y_0, Y_1, B_0, \dots, B_{b-1}$ are total multiconnection networks, then the multiconnection network shown in Figure 12 (which is slightly more general than the Clos network with $a=2$) is total.

Indeed, let f be a map $Z_{b_0+b_1} \rightarrow Z_{2b}$. In X_0 , we realize the map :

$$\xi_0 : k \rightarrow i \text{ if } k \in Z_{0,i}. \quad (17)$$

In X_1 , we realize the map :

$$\xi_1 : k \rightarrow b+i \text{ if } k \in Z_{1,i}. \quad (18)$$

In B_i ($i=0, \dots, b-1$), we realize the map β_i defined as follows :

$$\beta_i(0) = 0 \quad \text{if} \quad f(Z_{0,i}) = \{y_{0,i}\} \text{ or } \emptyset. \quad (19)$$

$$= 1 \quad \text{if} \quad f(Z_{0,i}) = \{y_{1,i}\} \quad (20)$$

$$\beta_i(1) = 1 \quad \text{if} \quad f(Z_{1,i}) = \{y_{1,i}\} \text{ or } \emptyset. \quad (21)$$

$$= 0 \quad \text{if} \quad f(Z_{1,i}) = \{y_{0,i}\}. \quad (22)$$

In \mathcal{V}_0 , we realize the map :

$$\eta_0 : i \rightarrow y_{0,i}. \quad (23)$$

In \mathcal{V}_1 , we realize the map :

$$\eta_1 : b+i \rightarrow y_{1,i}. \quad (24)$$

Then it is easy to check that the resulting map is f .

Now let us describe the algorithm for the determination of the maps ξ_0 , ξ_1 , η_0 , η_1 and β_i ($i=0,1,\dots,b-1$). As in Proposition 3, this is done by induction, and we get 5 cases, corresponding to the result (a) or (b) of Lemma 2, and to the cases (1°) to (4°) in the proof of it.

Suppose that for some $n \leq b$, all $Z_{j,i}$ and $y_{j,i}$ ($j \in Z_2$, $i \geq n$) have been determined.

In the first case, where $Y_j \cap \text{Im}f = \emptyset$ for some $j=0,1$, then all $Z_{j,i}$ and $y_{j,i}$ ($i < n$) can be determined as shown in Figure 13.

In the last four cases (illustrated in Figures 14 to 17), we determine $Z_0 = Z_{0,n-1}$, $Z_1 = Z_{1,n-1}$, $y_0 = y_{0,n-1}$ and $y_1 = y_{1,n-1}$, and we give the things which must be deleted in order to pass from n to $n-1$.

Let us give an example (with $b=4$).

Let us define f by the following matrix :

$$\begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 0 & 0 & 3 & 2 & 0 & 7 & 2 & 7 \end{pmatrix} \quad (25)$$

- (a) First we have $f^{-1}(0) = \{0,1,4\}$ and $f^{-1}(6) = \emptyset$. We apply then case 2 with $y=0$ and $y^1=6$. We delete 0,1 and 4 on the right, 0 and 6 on the left, and the box B_3 . Then f is reduced to a function g .
- (b) Then we have $g^{-1}(3) = \{2\}$ and $g^{-1}(7) = \{5,7\}$. We may thus apply case 4 with $y=3$ and $y^1=7$. We delete 2,5,7 on the right, 3 and 7 on the left, and the box B_2 . Now g is reduced to a function h .
- (c) Then we get $h^{-1}(2) = \{3,6\}$ and $h^{-1}(4) = \emptyset$. We apply case 2 with $y=2$ and $y^1=4$. We delete 3 and 6 on the right, 2 and 4 on the left, and the box B_1 . Then h is reduced to nothing, and we can set B_0 in any state and connect the outputs 0 of the two boxes on the left to any input.

This procedure is illustrated in Figure 18.

There are other networks which resemble to the Clos Network, for example Waksman's network [2]. M. Davio has shown that for $N=4$, this network, both in the direct and the reverse form, is not a total multiconnection network (see Figure 19).

D. Sorting networks as multiconnection networks

A sorting network is built with 2-cells with two states and decentralized control : each cell is equiped with a comparator which compares signals associated to its inputs and determines from it the state of the cell (see [1]).

We wish to use similar networks for multiconnection. First we use them in the reverse order (since we must realize a function from the outputs to the inputs, contrarily to what happens in a sorting network). Secondly we use 2-cells with more than two states (see Figure 9). We will use a multiconnection 2-cell with three states in the first case and four in the second one (these states are illustrated in Figure 20).

We illustrate the decentralized control in Figure 21. Here s_i is the signal associated to the output O_i ($i=0,1$). The comparator produces the signal s'_i ($i=0,1$) associated to the inputs I_i and determines the state of the cell.

Let us consider the first case. To every output O_y , we associate the signal

$$s(y) = f(y), \quad (26)$$

where f is the map that the network must realize. Each cell can have three states, as explained below :

- (1) $\underline{s_0 < s_1}$. Then we set $s'_1 = s_1$ and $s'_0 = s_0$, and switch the cell in the "straight" mode.
- (2) $\underline{s_0 > s_1}$. Then we set $s'_1 = s_0$ and $s'_0 = s_1$, and switch the cell in the "cross" mode.
- (3) $\underline{s_0 = s_1}$. Then we set $s'_0 = s'_1 = s_0 = s_1$ and switch the cell in the "down" mode.

Suppose that we have :

$$\text{Im}f = \{x_0, \dots, x_{k-1}\}, \quad (27)$$

where we set the x_i 's in increasing order, and for $i=0, \dots, k-1$, we write

$$F_i = f^{-1}(x_i) \quad (28)$$

and

$$b_i = |F_i|. \quad (29)$$

Then it follows from [3] that the network realizes the map g defined by

$$g(y) = 0 \quad \text{if } y \in F_0$$

$$= \sum_{j=0}^{i-1} b_j \quad \text{if } y \in F_j, \text{ where } 1 \leq j \leq k-1. \quad (30)$$

This is due to the fact that our network realizes in the reverse order a regular fusion-sorting (see [3]).

The procedure is illustrated in Figure 22 for $N=4$. The signals s_i are indicated by numbers affixed to the inputs and outputs of the cells.

Now if we look at our formulas (3), (4), (5) and (6), it is clear that the network realizes the same map as the two last stages of the three stage design.

Thus one can realize f by connecting a permutation network to the inputs of our network. If we want decentralized control, then we can choose for it a sorting network (built from 2-cells with 2 states) set in the reverse order.

Let us now introduce the second case :

One of the drawbacks of the preceding network is that we need to connect two cellular networks derived from sorting networks in order to be able to realize all multiconnections.

One might wonder whether one could have a better result by choosing multiconnection cells with 4 states (as in Figure 20) instead of 3 and 2 respectively. The problem reduces then to choosing the correct O_y asso-

ciated to O_y and the behavior of the multiconnection cell in function of its signals s_0 and s_1 associated to its outputs.

Suppose that we choose to take $s(y) = f(y)$ and set the cells in the following modes :

- If $s_1 > s_0$, then $s'_1 = s_1$ and $s'_0 = s_0$, and the cell is in the straight mode.
- If $s_1 < s_0$, then $s'_0 = s_1$ and $s'_1 = s_0$, and the cell is in the cross mode.
- If $s_1 = s_0$, then we choose s'_0 and s'_1 according to our convenience, and the cell is in the up or down mode.

(This is the most general extension of the case where f is a permutation and where we use the normal sorting procedure).

Then the network does not always realize f . We give two examples in Figure 23, where f is impossible to realize and where we get instead two maps f_1 and f_2 .

Thus we have to choose another method. We will define the fictive permutation method, which works only for certain types of sorting networks.

Again, we use (27), (28) and (29). In each F_i , we choose an element y_i , called the representative of F_i .

We call then a fictive permutation associated to f and the representatives y_0, \dots, y_{k-1} , any permutation g of Z_N such that

$$g(y_i) = f(y_i) \text{ for } i=0, \dots, k-1 \quad (31)$$

We will now choose our associated signals in $Z_N \times Z_N$. Indeed, we associate to every output O_y the signal

$$s(y) = (g(y), f(y)). \quad (32)$$

For any distinct $y, y' \in Z_N$, we have one of the following three possibilities

(i) $y \in F_i, y' \in F_i$, and either $i \neq i'$ or $i = i'$ and $y \neq y_i \neq y'$.

Then $f(y) = g(y_i), f(y') = g(y_i), g(y_i) \neq g(y')$ and $g(y_i) \neq g(y)$. Thus we have :

$$\begin{aligned} f(y) &\neq g(y') \\ \text{and } f(y') &\neq g(y). \end{aligned} \quad (33)$$

(ii) $y = y_i, y' \in F_i \setminus \{y_i\}$ and then :

$$\begin{aligned} f(y) &\neq g(y') \\ \text{and } f(y') &= g(y), \end{aligned} \quad (34)$$

since $f(y) = g(y), g(y') \neq g(y)$ and $f(y') = f(y)$.

(iii) $y' = y_i, y \in F_i \setminus \{y_i\}$ and then we get by symmetry with (ii) :

$$\begin{aligned} f(y) &= g(y') \\ \text{and } f'(y) &\neq g(y). \end{aligned} \quad (35)$$

As (33), (34) and (35) are mutually exclusive, they are equivalent to (i), (ii) and (iii) respectively.

Suppose now that we perform on the network the sorting in relation with g , in other words if we set each cell in the following state (where $s_i = (g_i, f_i)$ for $i=0,1$) :

- straight for $g_0 < g_1$.
- cross for $g_0 > g_1$

and if we do the same for the determination of the s_i' ($i=0,1$), then the network will realize g .

Now suppose that we can choose g in such a way that for any $i=0, \dots, k-1$ and for any $y \in F_i \setminus \{y_i\}$, the paths $O_y \rightarrow I_{g(y)}$ and $O_{y_i} \rightarrow I_{g(y_i)}$ pass at least once through a common cell. Then we change the connection from the output of the cell corresponding to O_y and we connect it to the same input as the output of the cell corresponding to O_{y_i} . Then O_y will be connected to $I_{g(y_i)} = I_{f(y)}$. It follows that the network will realize f .

This transformation is illustrated in Figure 24.

Of course this possibility is not guaranteed for every type of sorting network. For example the network of Figure 22 cannot in any way realize the map

$$\begin{aligned} f : 0 &\rightarrow 0 \\ 1 &\rightarrow 0 \\ 2 &\rightarrow 0 \\ 3 &\rightarrow 2 \end{aligned} \tag{36}$$

whether with a centralized or a decentralized control. However, it is a sorting network set backwards.

Now let us give the input-output behavior of the cells shown in Figure 2. We suppose that we have $s_i = (g_i, f_i)$.

First, for every $y \in Z_n$, the signal associated to it is (32).

For any relation F , we define the truth function $\delta(F)$ by :

$$\begin{aligned} \delta(F) &= 1 \text{ if } F \text{ is true.} \\ &= 0 \text{ if } F \text{ is false.} \end{aligned} \tag{37}$$

We have thus the following two tables :

$\delta(g_0 > g_1)$	s'_0	s'_1
0	s_0	s_1
1	s_1	s_0

(38)

$\delta(g_0 > g_1)$	$\delta(g_0 = f_1)$	$\delta(g_1 = f_0)$	Cell mode
0	0	0	straight
1	0	0	cross
0	1	0	down
1	1	0	up
0	0	1	up
1	0	1	down

(39)

The first table is obvious. Let us explain the second one. Suppose that $s_0 = s(y)$ and $s_1 = s(y')$ for some $y, y' \in Z_N$. In the first two lines, (33) holds, and so the cell must be in the sorting mode. Thus one get "straight" for $g_1 > g_0$ and "cross" for $g_0 > g_1$. In the next two lines, (34) holds, and so $y = y_i$ and $y' \in F_i \setminus \{y_i\}$ for some $i=0, \dots, k-1$. We must then change the input connected to O_1 . Thus "straight" becomes "down" and "cross" becomes "up". In the last two lines, (35) holds, and so $y' = y_i$ and $y \in F_i \setminus \{y_i\}$ for some i . We must then change the input connected to O_0 . Thus "straight" becomes "up" and "cross" becomes "down".

Suppose that the multiconnection 2-cell is built as in Figure 9, using two binary variable c_0 and c_1 . Then the 4 states are given in the following table :

c_1	c_0	state
0	0	straight
0	1	cross
1	0	down
1	1	up

(40)

Then the inner control of the 2-cell (see Figure 21) is illustrated in Figure 25, where "P.N." means "permutation network" and "B.N." means "branching network", and where Φ is a switching network realizing the input-output behavior :

$$(\delta(g_0 > g_1), \delta(g_1=f_0), \delta(g_0=f_1)) \rightarrow (c_1, c_0). \quad (41)$$

From (39) and (40) we have :

$$c_1 = \delta(g_1=f_0) \vee \delta(g_0=f_1) \quad (42)$$

and

$$c_0 = \delta(g_0 > g_1) \oplus \delta(g_1=f_0). \quad (43)$$

As we said above, it is not guaranteed that we can choose g in such a way that it satisfies the condition on the paths $0_y \rightarrow I_{g(y)}$, where $y \in F_i \setminus \{y_i\}$. When g satisfies that condition, we say that g is a suitable (fictive bijection).

Let us make the following definition :

Let N be a switching network having n inputs and n outputs ($n > 0$), which is built with 2-cells, i.e. smaller networks with 2 inputs and 2 outputs. Then we say that N is adjacent if we can decompose N in r stages ($r \geq 1$) N_0, \dots, N_{r-1} having n inputs and n outputs each, such that :

- (i) For every $z \in Z_n$ and for any $i=0, \dots, r-2$, the output z of N_i is connected to the input z of N_{i+1} .
- (ii) For each $i=0, \dots, r-1$, the stage N_i consists of :
 - straight connections $I_z \rightarrow O_z$ ($z \in Z_n$), or
 - 2-cells with inputs I_z, I_{z+1} and outputs O_z, O_{z+1} ($z \in Z_{n-1}$), where I_z and O_z represent the input z and the output z ($z \in Z_n$) of N_i .

We give an example in Figure 26.

We have the following result :

Lemma 4. Let N be a multiconnection network with N inputs and N outputs built from an adjacent sorting network. Let f be a map $Z_N \rightarrow Z_N$. Let g be a fictive permutation associated to f . Then a sufficient condition for g to be suitable is that :

$$\begin{aligned} & \text{For any } i=0, \dots, k-1 \text{ and } y \in F_i \setminus \{y_i\}, \\ & g(y) < g(y_i) \text{ if and only if } y > y_i. \end{aligned} \quad (44)$$

Proof. As the network is adjacent, we can suppose that there are N levels $0, \dots, N-1$ and that a binary cell connects each of its output to an input on the same or an adjacent level. Suppose that (44) holds. We have the r stages of the definition, going from the outputs to the inputs (since

we have taken the sorting network in the reverse order). Let j be the highest integer such that after passing through all the stages N_u ($u < j$) the paths starting from O_y and O_{y_i} are at levels L_y and L_{y_i} such that

$$L_{y_i} < L_y \quad \text{iff} \quad y_i < y. \quad (45)$$

Then clearly $0 \leq u < r$ by (44). Thus there is a stage N_j . After that stage, we pass to levels L'_{y_i} and L'_y such that :

$$\begin{aligned} L'_{y_i} < L'_y & \quad \text{iff} \quad y_i > y \\ & \quad \text{iff} \quad L_{y_i} > L_y \end{aligned} \quad (46)$$

As the network is adjacent, we have :

$$0 \leq |L'_{y_i} - L_y|, |L_{y_i} - L'_y| \leq 1 \quad (47)$$

From (46) and (47) we get :

$$\begin{aligned} |L_{y_i} - L_y| &= 1, \\ L_{y_i} &= L'_y, \\ \text{and } L_y &= L'_{y_i} \end{aligned} \quad (48)$$

This means that the paths from O_{y_i} and O_y pass through a common cell. Thus g is suitable.

We have then the following :

Lemma 5. (M. Davio). For any map $f : Z_N \rightarrow Z_N$, there is a fictive bijection g satisfying condition (44).

Proof. We determine $g(y)$ for $y \in F_i$ by induction on i . Suppose that for any $j < i$ and $y \in F_j$, $g(y)$ is determined (this condition is trivial for $i=0$) in such a way that $g(y) \neq f(z)$ for $z \in F_r$, $r > j$.

Define :

$$T = \left\{ x \in Z_N \mid \begin{array}{l} x \neq g(y) \text{ for any } y \in F_j, j < i \\ x \neq f(z) \text{ for any } z \in F_r, r > i \end{array} \right\} \quad (49)$$

We have by (27), (28) and (29) :

$$\begin{aligned} |T| &= N - \sum_{j < i} b_j - \left| \bigcup_{r > i} f(F_r) \right| \\ &\geq N - \sum_{j < i} b_j - \sum_{r > i} b_r = b_i \end{aligned} \quad (50)$$

We take a subset T_i of T such that $|T_i| = b_i$ and $x_i \in T_i$ (see (27)).

We can write :

$$F_i = \{y_1^{(i)}, \dots, y_{b_i}^{(i)}\},$$

with

$$y_u^{(i)} < y_v^{(i)} \text{ for } u < v, \quad (51)$$

and

$$T_i = \{z_1^{(i)}, \dots, z_{b_i}^{(i)}\},$$

with

$$z_u^{(i)} > z_v^{(i)} \text{ for } u < v. \quad (52)$$

Then we define for $u=1, \dots, b_i$:

$$g(y_u^{(i)}) = z_u^{(i)}. \quad (53)$$

If y_i is the element of F_i mapped by g on x_i , then y_i satisfies (44) for any $y \in F_i \setminus \{y_i\}$.

Thus for any $y \in F_i$, $g(y)$ is determined and $g(y) \neq f(z)$ for $z \in F_r$, $r > i$. Thus this construction can be applied to F_{i+1}, \dots, F_{k-1} , and so the result holds.

This construction is illustrated in Figure 27 for $N=6$.

Two well-known adjacent sorting networks are (see [1,2]) :

- the diamond array, which exists in two forms (similar up to a rotation of 180°) and is illustrated in Figure 28 for even and odd n .
- the triangular array (see Figure 29).

In Figure 30, we apply our algorithm to the triangular network for $N=5$. The control signals $s_i = (g_i, f_i)$ are written $g_i f_i$, i.e. as number in radix 5. We write them at the inputs and outputs of the 2-cells.

Although this algorithm has decentralized control, it requires a centralized operation, i.e. the determination of a suitable fictive permutation. The algorithm used in Lemma 5 and illustrated in Figure 27 has cost of the form $O(N^2)$. Note that the cost of the diamond and triangular arrays in terms of 2-cells is $N(N-1)/2$, which is also of the form $O(N^2)$.

III. Complexity.

We will study the cost, delay and complexity of the control algorithm in total multiconnection networks. We will limit ourselves to square networks (i.e. with $M=N$).

A. Theoretical complexity bounds

Consider a total multiconnection network S on N bits. It has N^N distinct states. If the control inputs use m -ary logic ($m \geq 2$), then

$N \log_m(N)$ control inputs are needed. Now we count the cost of S in terms of logical gates of bounded fan-in. As the sum of all fan-ins of these gates includes the $N \log_m(N)$ control inputs, and as the fan-in of every gate is bounded by a number b , there are at least $\frac{1}{b} N \log_m(N)$ gates. Thus the cost is at least proportional to $N \log(N)$. Now the three-stage network has cost asymptotically equal to $N \log_2(N) (2\gamma + \gamma^1)$, where γ is the cost of a permutation 2-cell and γ^1 is the cost of a branching 2-cell, if one uses Green's permutation network (see [2]) and the second branching network described above, which is illustrated in Figure 8.

Thus the minimal cost of an N bit total multiconnection network is asymptotically proportional to $N \log(N)$.

If we build our network with cells, then an argument similar to Proposition 0 of [2] shows that the delay of S is at least proportional to $N \log N$. Now the three stage network using the components described above has delay asymptotically proportional to $\log_2(N) (4\delta + 2\delta^1)$.

Thus the minimal delay of an N bit total multiconnection network is asymptotically proportional to $\log(N)$.

It is easily shown that a branching network on N bits has cost at least $N-1$ and delay at least $\log_2(N)$ in terms of branching 2-cells. The bound is attained by the first type of branching network (illustrated in Figure 7), while the delay bound is asymptotically half the value attained by the delay of the second type of branching network (illustrated in Figure 8).

B. Comparison of the different designs

The direct design has cost proportional to N^2 , and its control algorithm is very easy. If the multiplexers are built with multiplexers on 2 bits, then the delay is in $\log(N)$.

The three stage network using Green's network [2] as permutation network has cost and delay of the types $N \log(N)$ and $\log(N)$ as seen above, but the control algorithm of the permutation network is very costly.

If one uses the first type of branching network and diamond networks as permutation networks in the three stage network, then one get a very easy control algorithm (it is completely decentralized), but the cost and delay are then asymptotically N^2 and $3N$ in term of 2-cells.

If one uses the Clos network with $a=2$ in iteration, then such a network built from 2-cells has asymptotic cost and delay equal to $\frac{1}{2} N^2$ and $2N$ respectively. Moreover, its control algorithm is very difficult.

The two designs of section II.C. have the following asymptotic cost and delay in terms of 2-cells if one uses diamond networks as sorting networks

$$\begin{array}{ll} - N^2 & \text{and } 2N \\ - \frac{1}{2} N^2 & \text{and } N. \end{array}$$

The first one has trivial decentralized control. The second one uses decentralized control but requires a preliminary operation costing N^2 .

Conclusion. For cheapness in cost and delay it is better to take the three stage design using Green's network. For cheapness of the control algorithm, it is better to take the three stage design using a sorting network, or the first design based on sorting networks of section II.C. The Clos network is very impractical, but it is the only design for the expansion of total multiconnection networks (a problem which one encounters directly because of the limited size of microprocessors).

REFERENCES

- [1] D.E. Knuth : "Sorting and Searching". Vol. 3 of "The Art of Computer Programming", Addison-Wesley, Reading Mass., 1973.
- [2] C. Ronse : "Cellular Permutation Networks : A Survey". Report R415, PRLB, December 1979.
- [3] C. Ronse : "Networks for Sorting with Fusion". Report R446, PRLB, November 1980.
- [4] S. Sakata, H. Hirayama, Y. Shibata : "Synthesis of Multiconnected Switching Networks". Electronics and Communications in Japan, Vol. 58-A, n° 1, 1975.

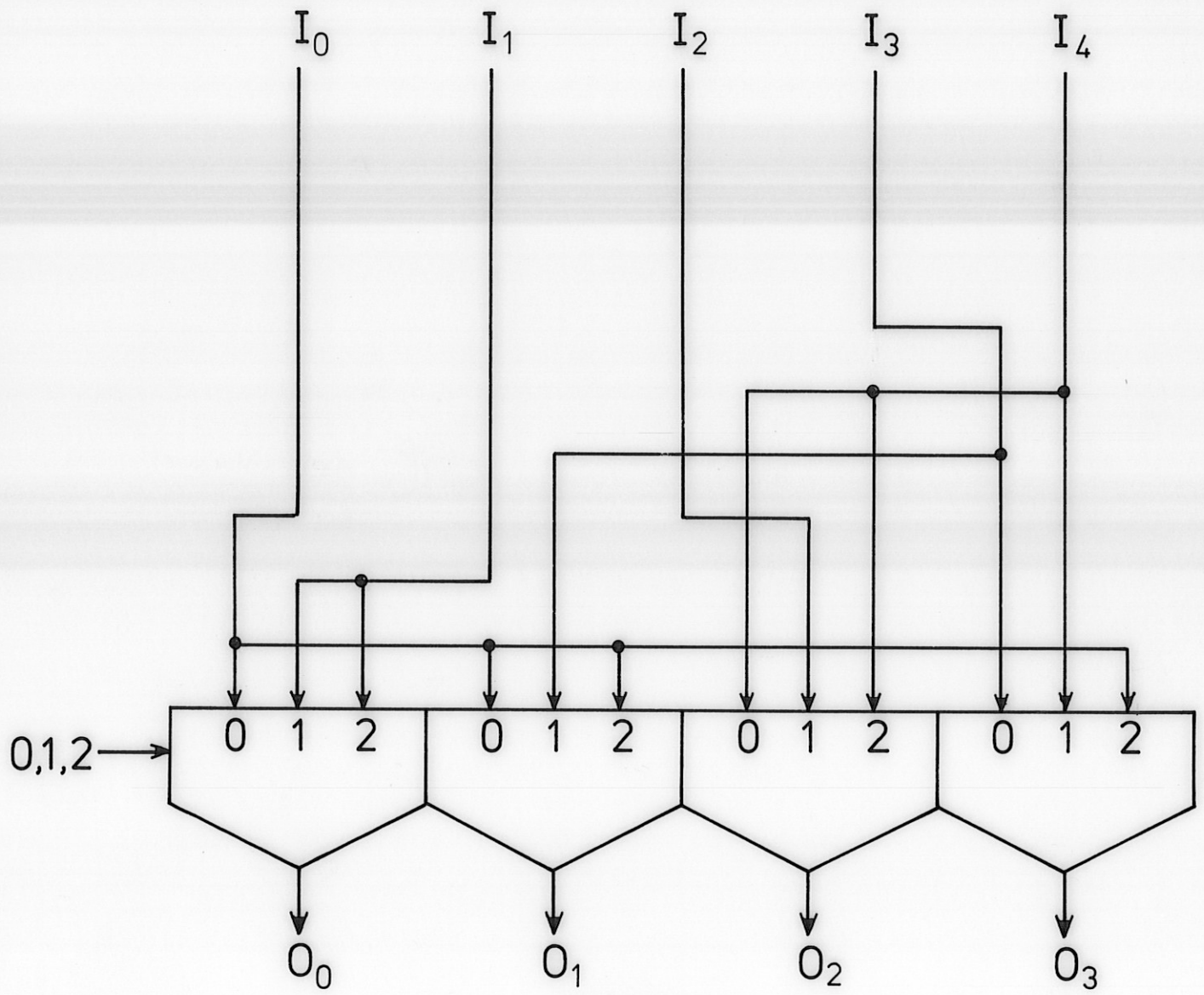


Figure 1.

$\underline{f_0}$
 $0 \rightarrow 0$
 $1 \rightarrow 0$
 $2 \rightarrow 4$
 $3 \rightarrow 3$

$\underline{f_1}$
 $0 \rightarrow 1$
 $1 \rightarrow 3$
 $2 \rightarrow 2$
 $3 \rightarrow 4$

$\underline{f_2}$
 $0 \rightarrow 1$
 $1 \rightarrow 0$
 $2 \rightarrow 4$
 $3 \rightarrow 0$

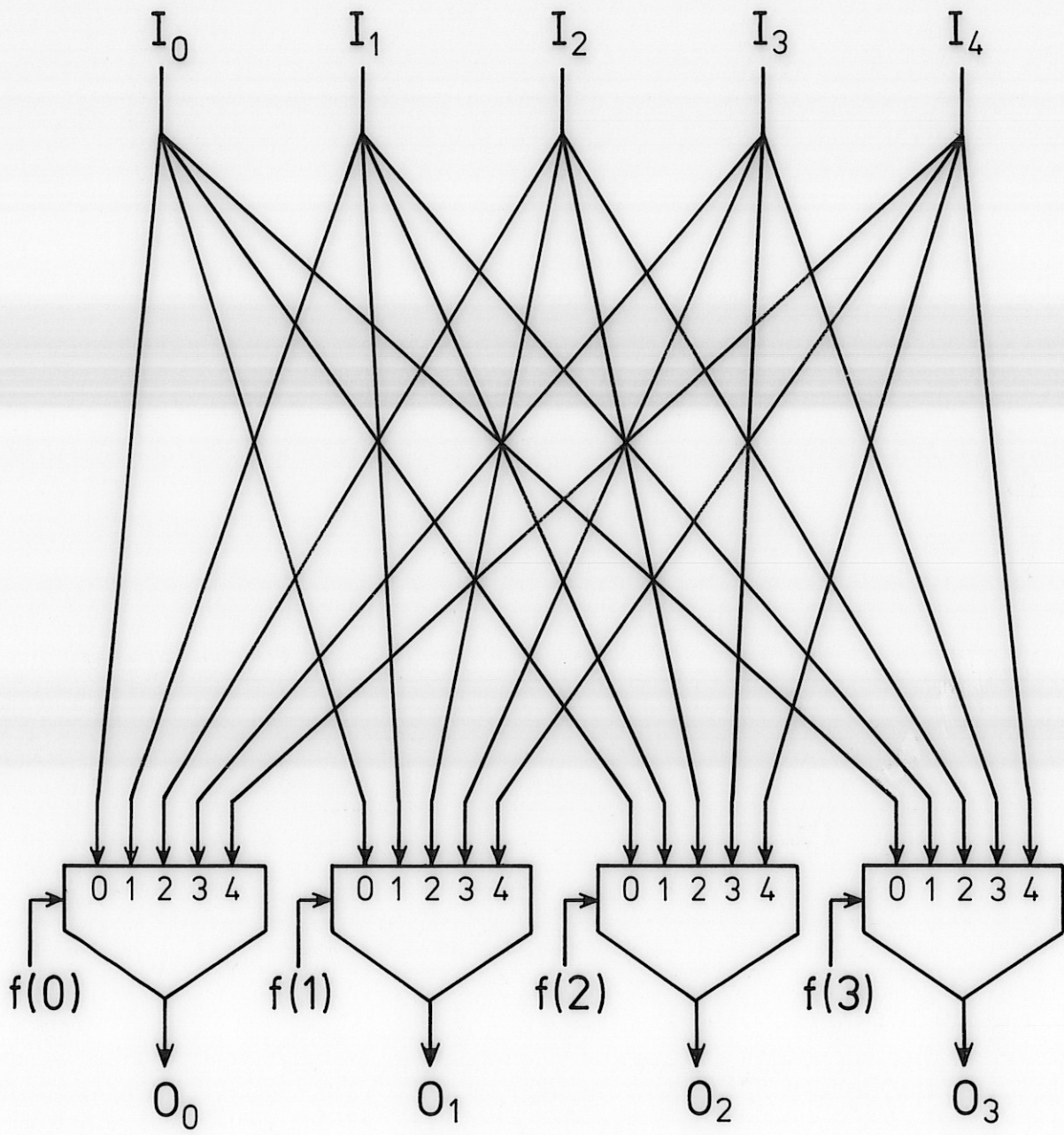


Figure 2.

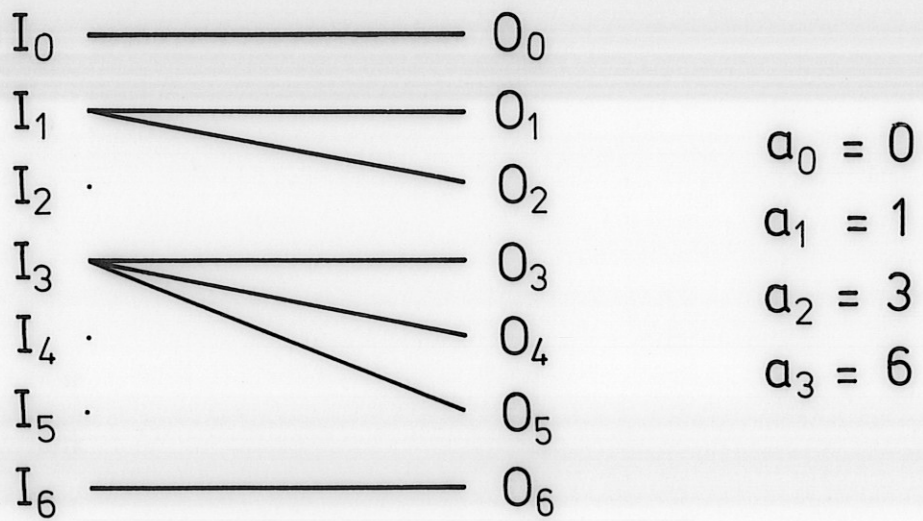
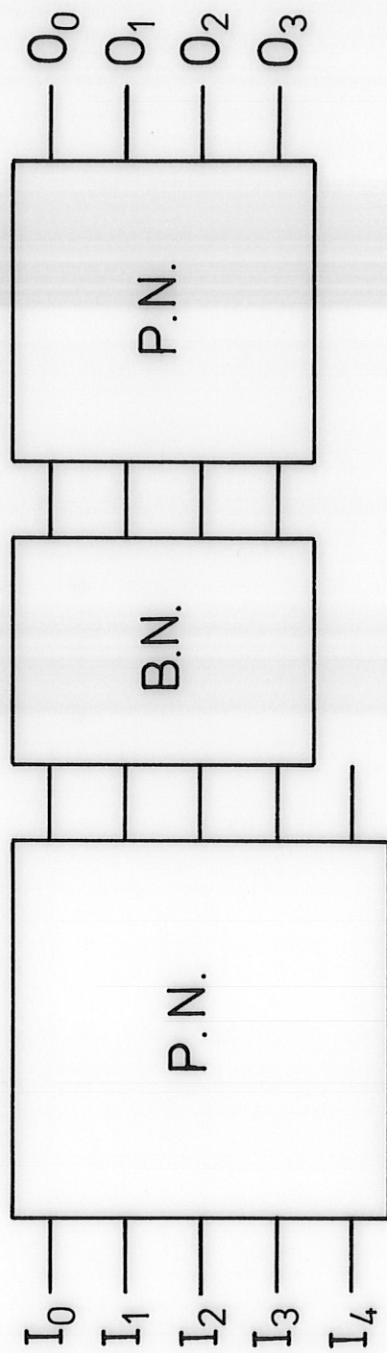


Figure 3.

A branching : $N=7$, $k=4$.



P.N. = Permutation Network.

B.N. = Branching Network.

Figure 4.

The three stage network : $N=4$, $M=5$.

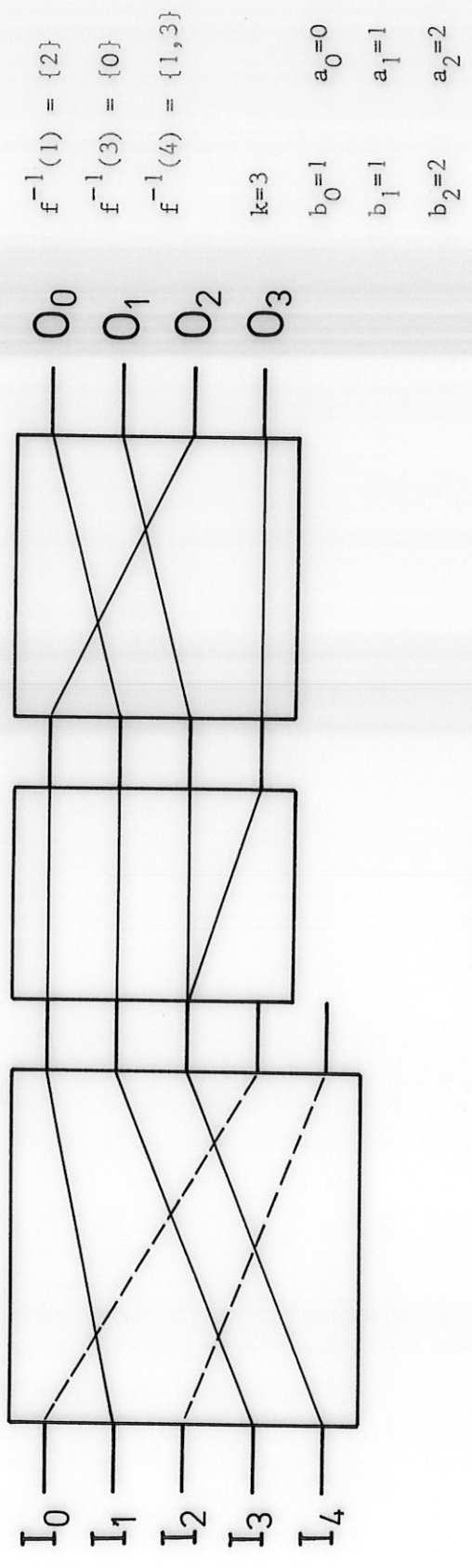
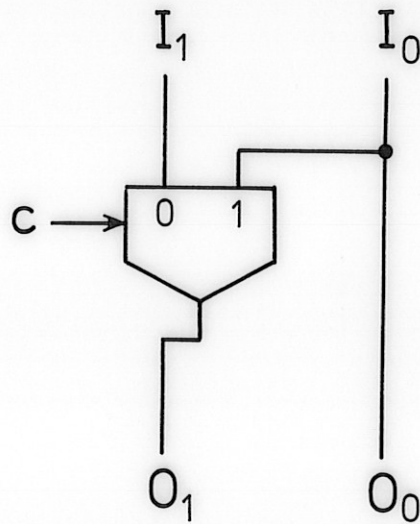
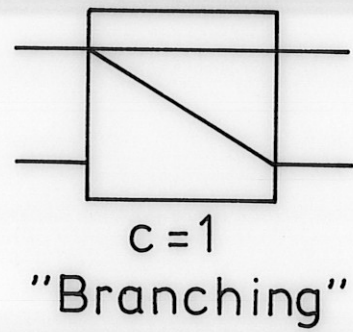
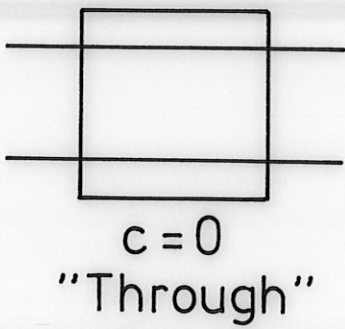
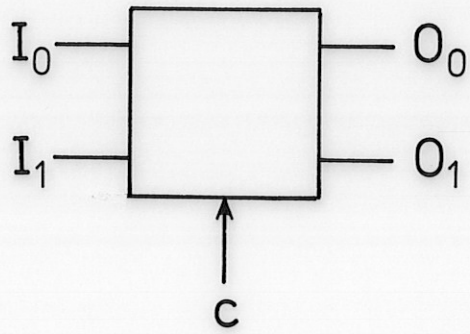


Figure 5.



Design using a multiplexer.

Figure 6.

A branching cell with its 2 states.

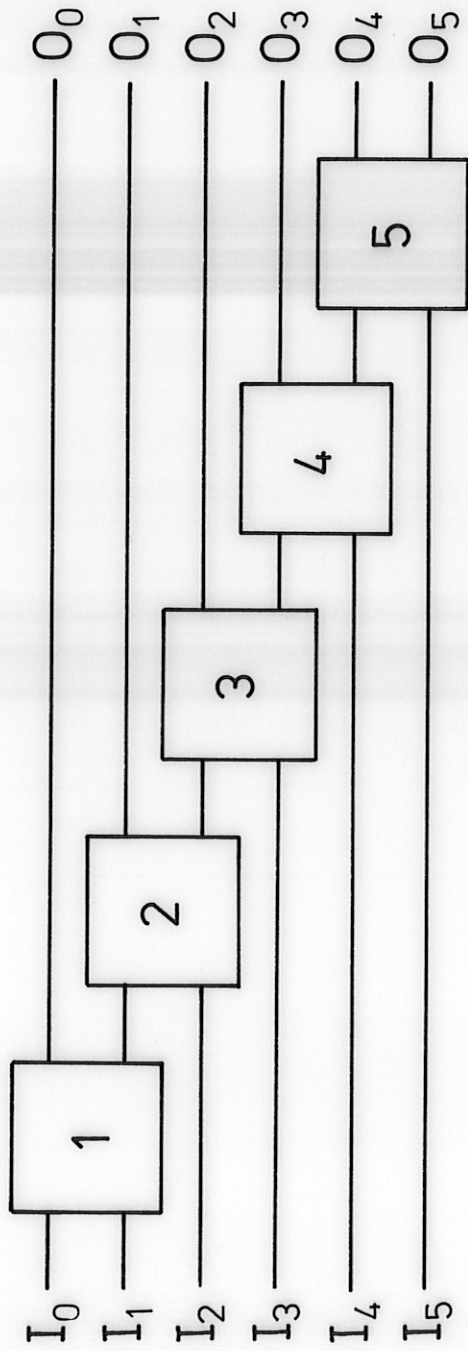


Figure 7. A branching network $N=6$.

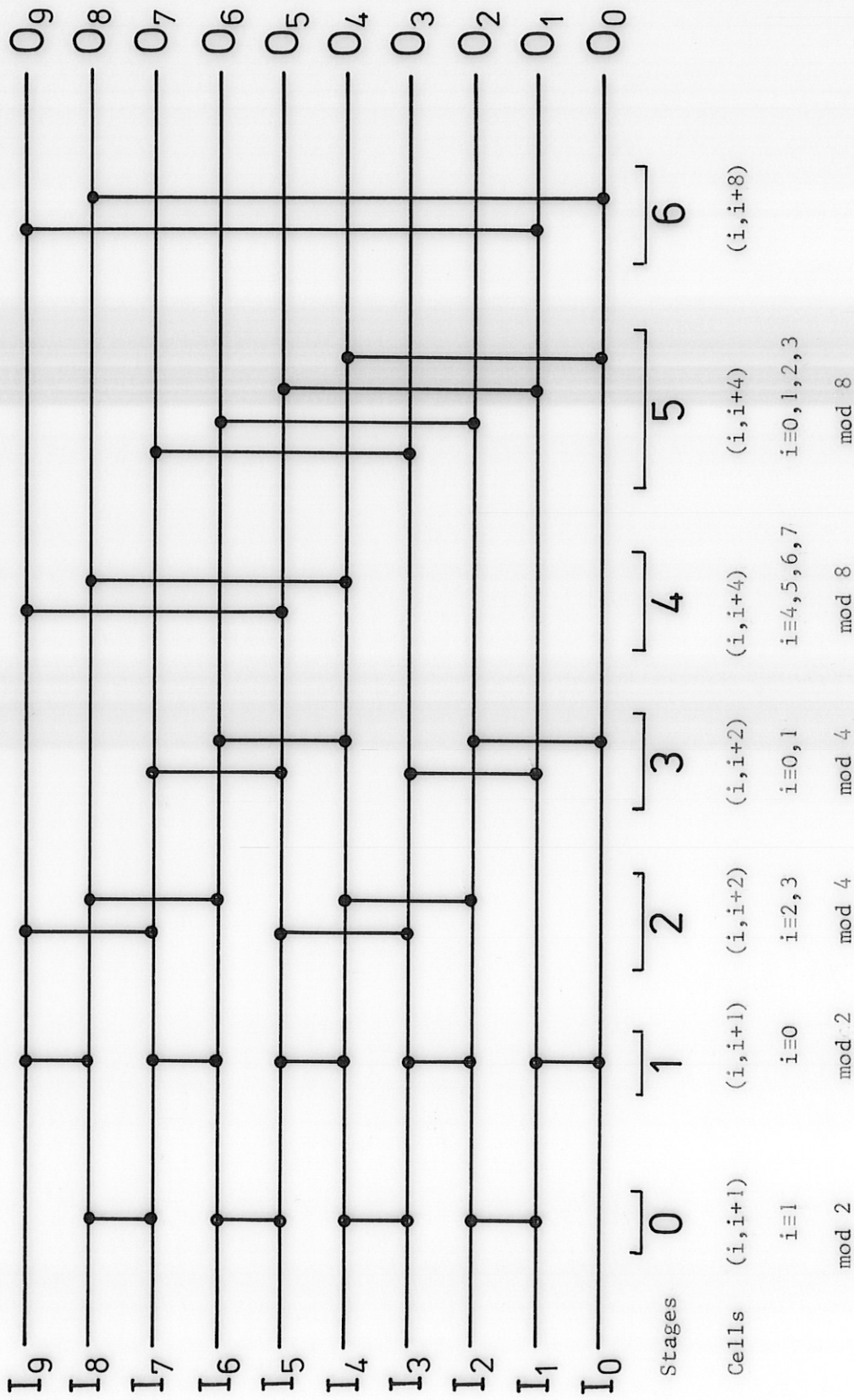

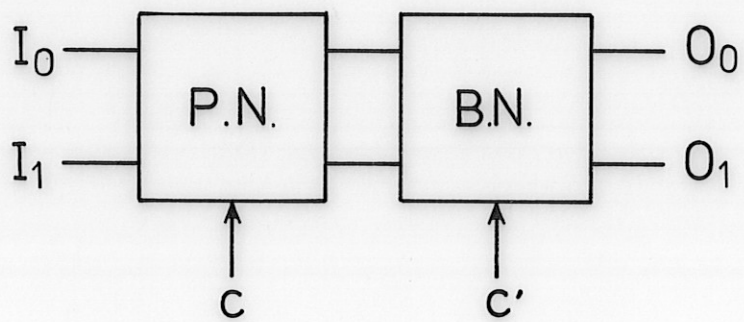


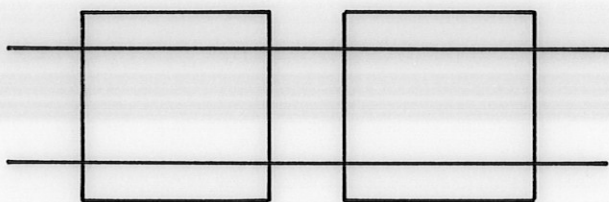
Figure 8. A branching network.


 = branching cell

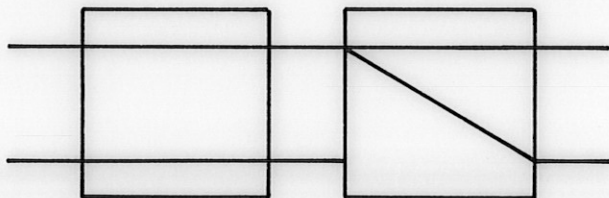
$N=10$
 $(n=4)$



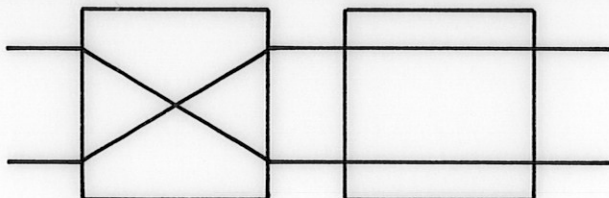
P.N. = Permutation
Network
B.N. = Branching
Network.



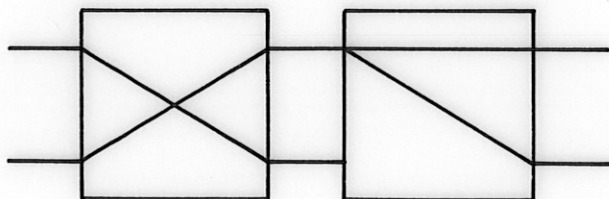
$$c = c' = 0$$



$$c = 0, c' = 1$$



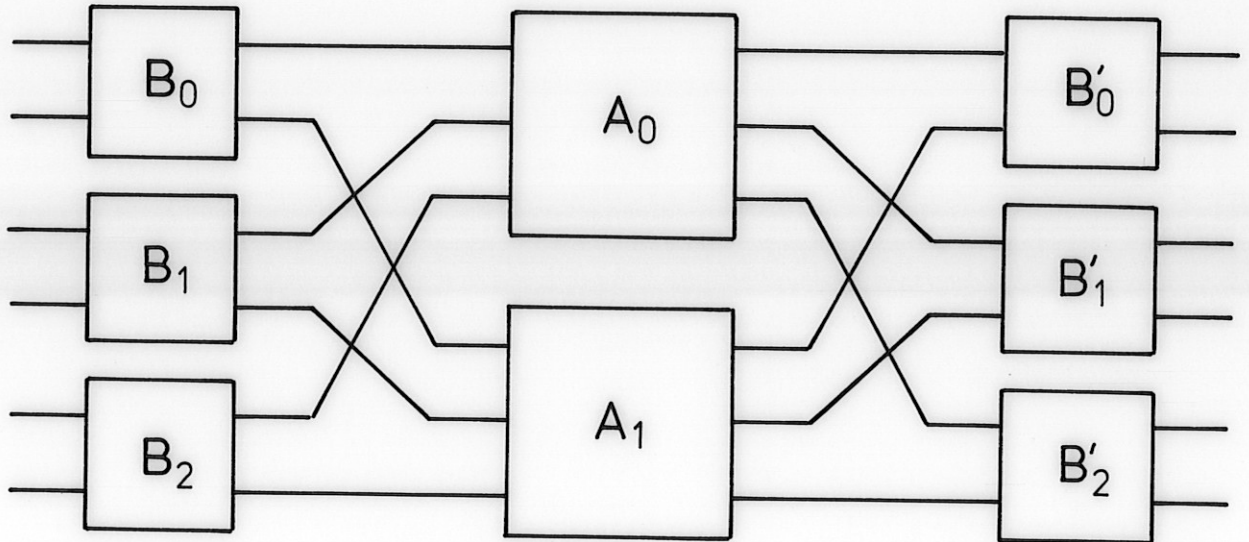
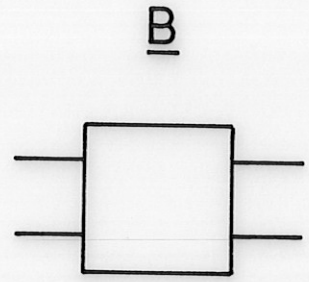
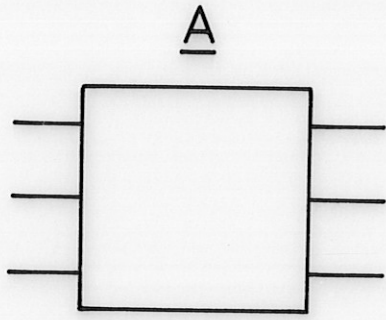
$$c = 1, c' = 0$$



$$c = c' = 1$$

Figure 9.

A total multiconnection network for $N=2$ and its 4 states



$B \times A$

Figure 10.

The Clos Network.

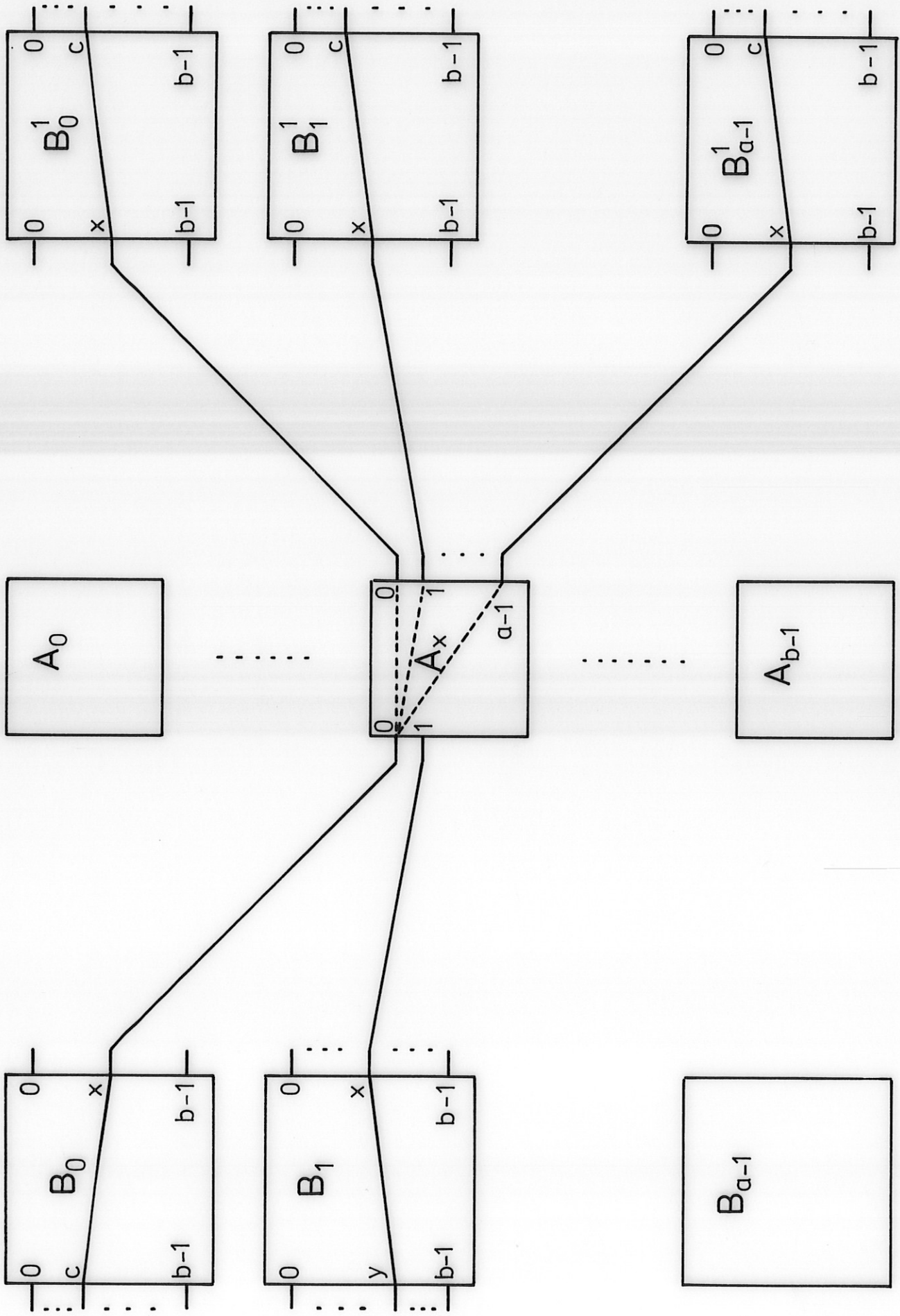


Figure 11.

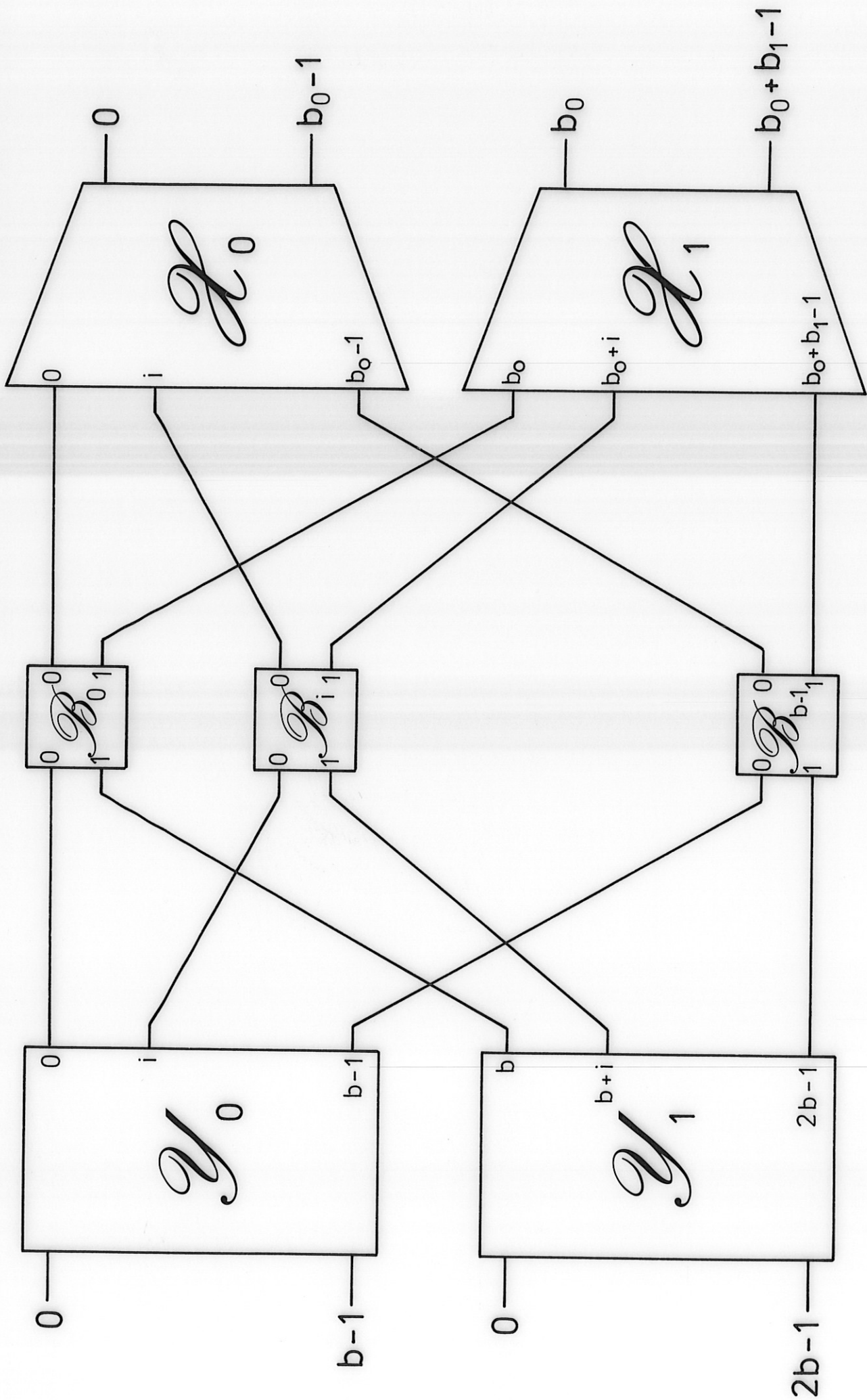
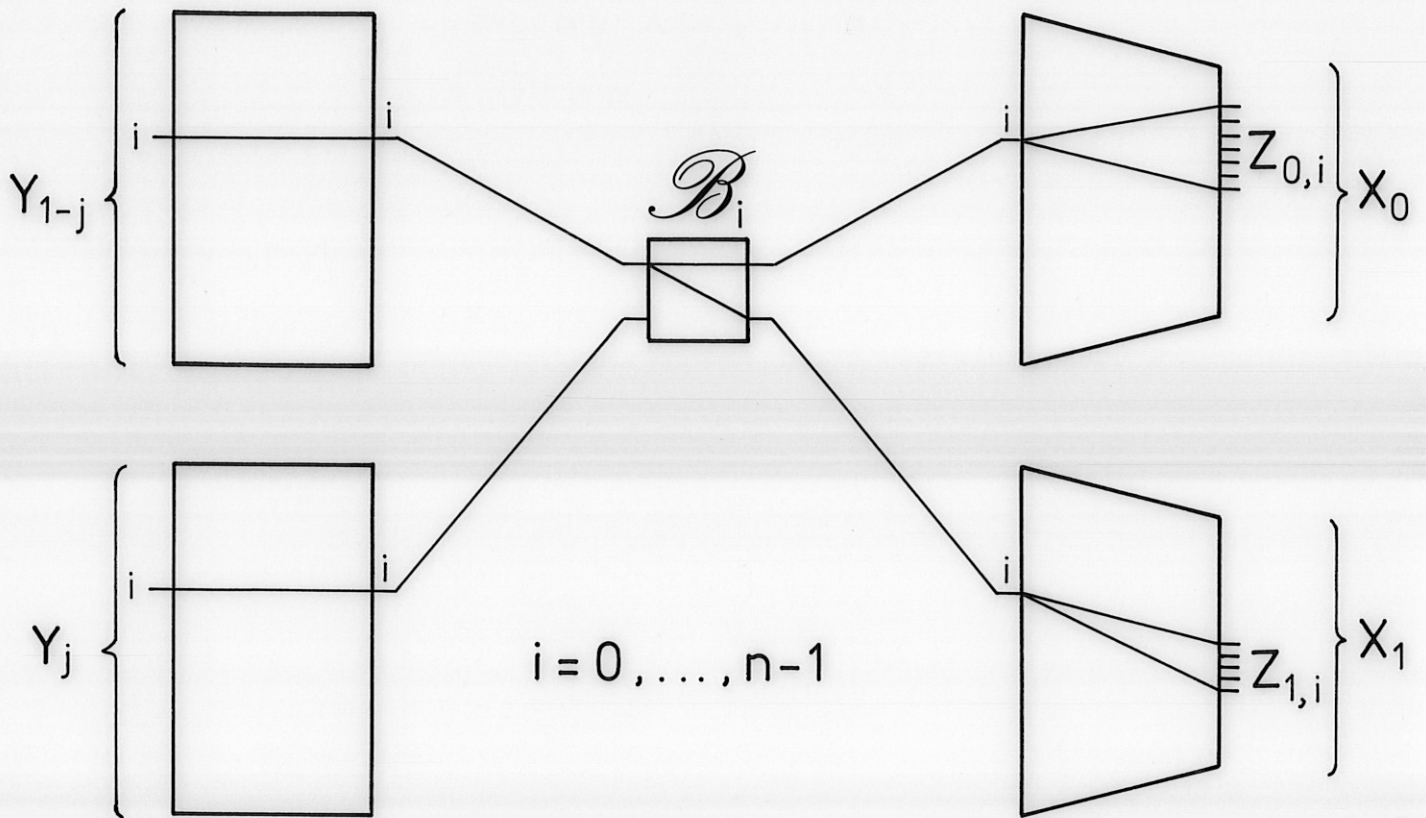


Figure 12. $b_0, b_1 \leq b$.

Case 1. $Y_j \cap \text{Im}f = \emptyset$



$y_{0,i}$ = i th element of Y_0 .

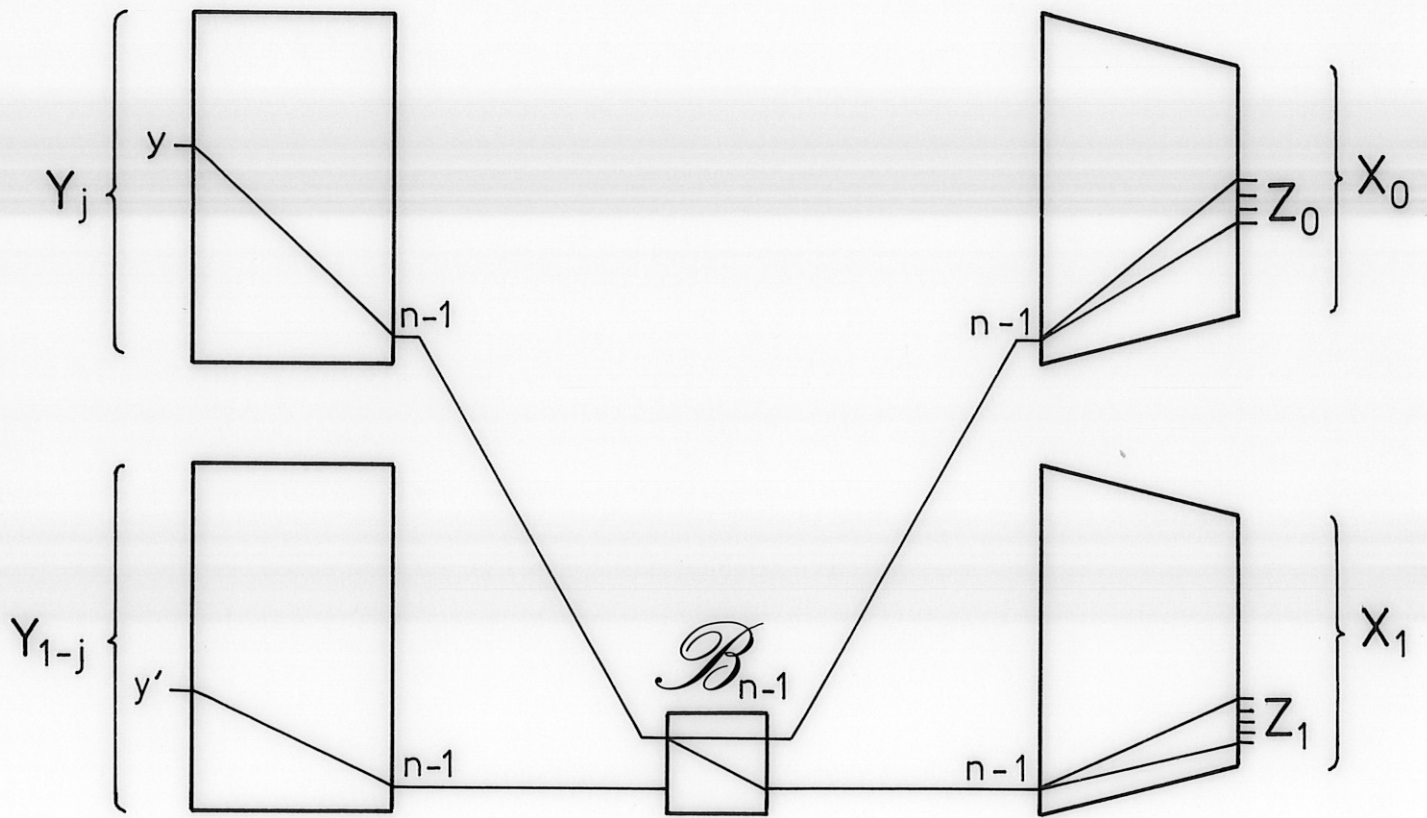
$y_{1,i}$ = i th element of Y_1 .

$Z_{0,i}$ = $f^{-1}(y_{1-j,i}) \cap X_0$.

$Z_{1,i}$ = $f^{-1}(y_{1-j,i}) \cap X_1$.

Figure 13.

Case 2. $y \in Y_j$, $y' \in Y_{1-j}$.
 $f^{-1}(y) \cap X_0 \neq \emptyset \neq f^{-1}(y) \cap X_1$.
 $f^{-1}(y') = \emptyset$.



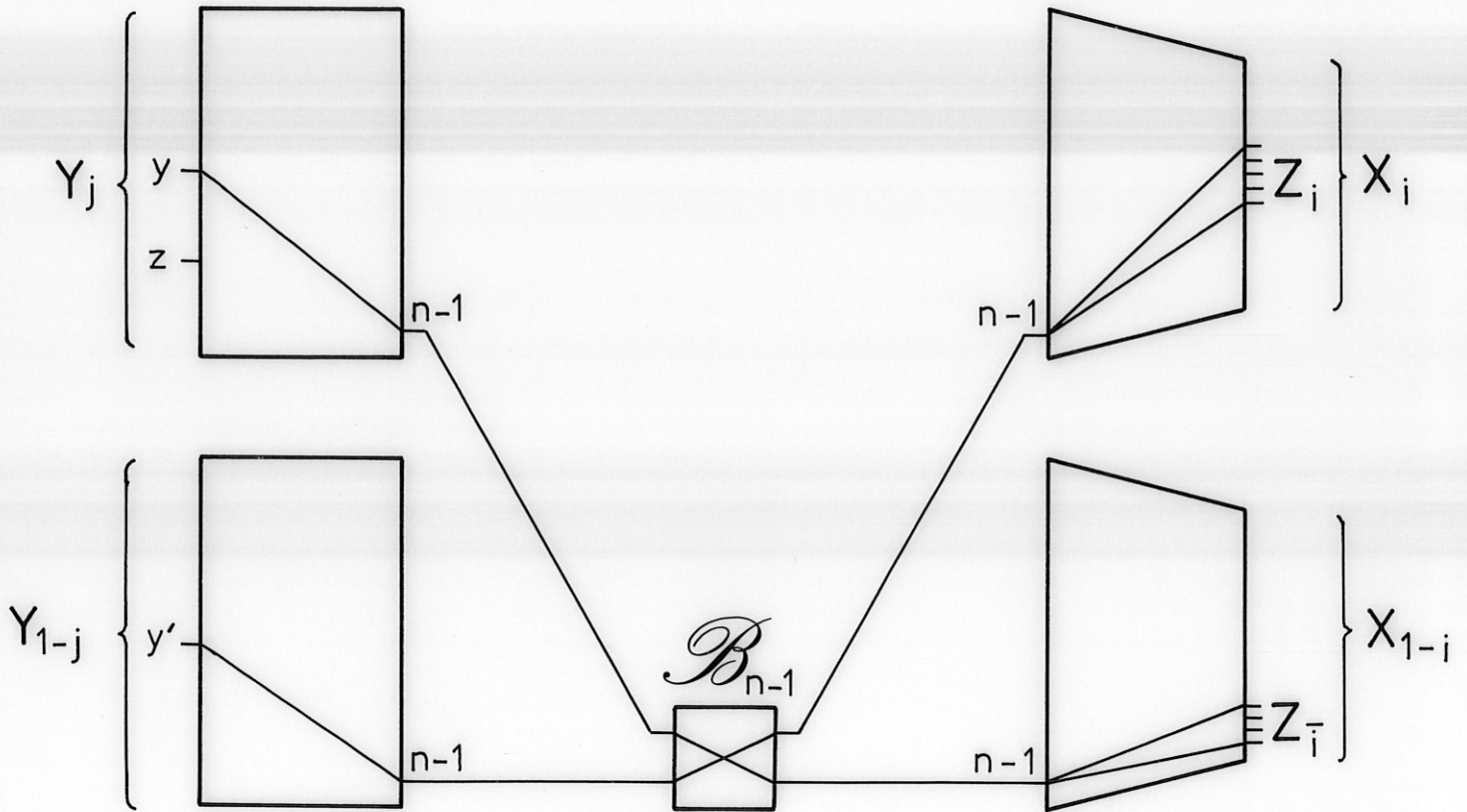
$$Z_0 = f^{-1}(y) \cap X_0 .$$

$$Z_1 = f^{-1}(y) \cap X_1 .$$

Delete y, y', Z_0, Z_1, B_{n-1} .

Figure 14.

Case 3. $y, z \in Y_j$, $y' \in Y_{1-j}$.
 $f^{-1}(y) \cap X_0 \neq \emptyset \neq f^{-1}(y) \cap X_1$.
 $f^{-1}(z) = \emptyset$.
 $\emptyset \neq f^{-1}(y') \subseteq X_i$.



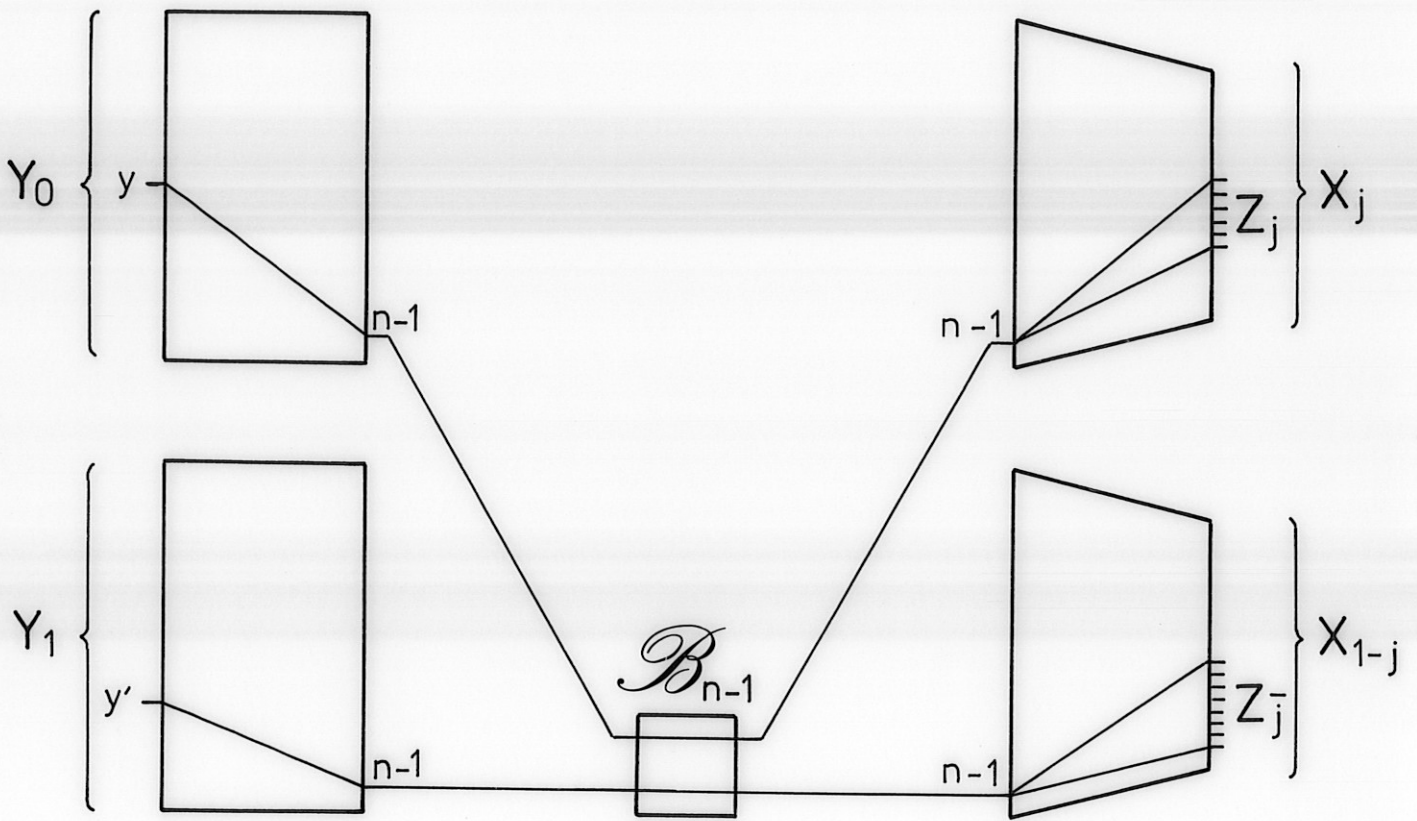
$$Z_i = f^{-1}(y') .$$

$$Z_{1-i} = f^{-1}(y) \cap X_{1-i} .$$

Delete z, y', Z_0, Z_1, B_{n-1}

Figure 15.

Case 4. $y \in Y_0$, $y' \in Y_1$
 $\emptyset \neq f^{-1}(y) \subseteq X_j$
 $\emptyset \neq f^{-1}(y') \subseteq X_{1-j}$



$z_j = f^{-1}(y)$
 $z_{1-j} = f^{-1}(y')$
Delete y, y', z_0, z_1, B_{n-1} .

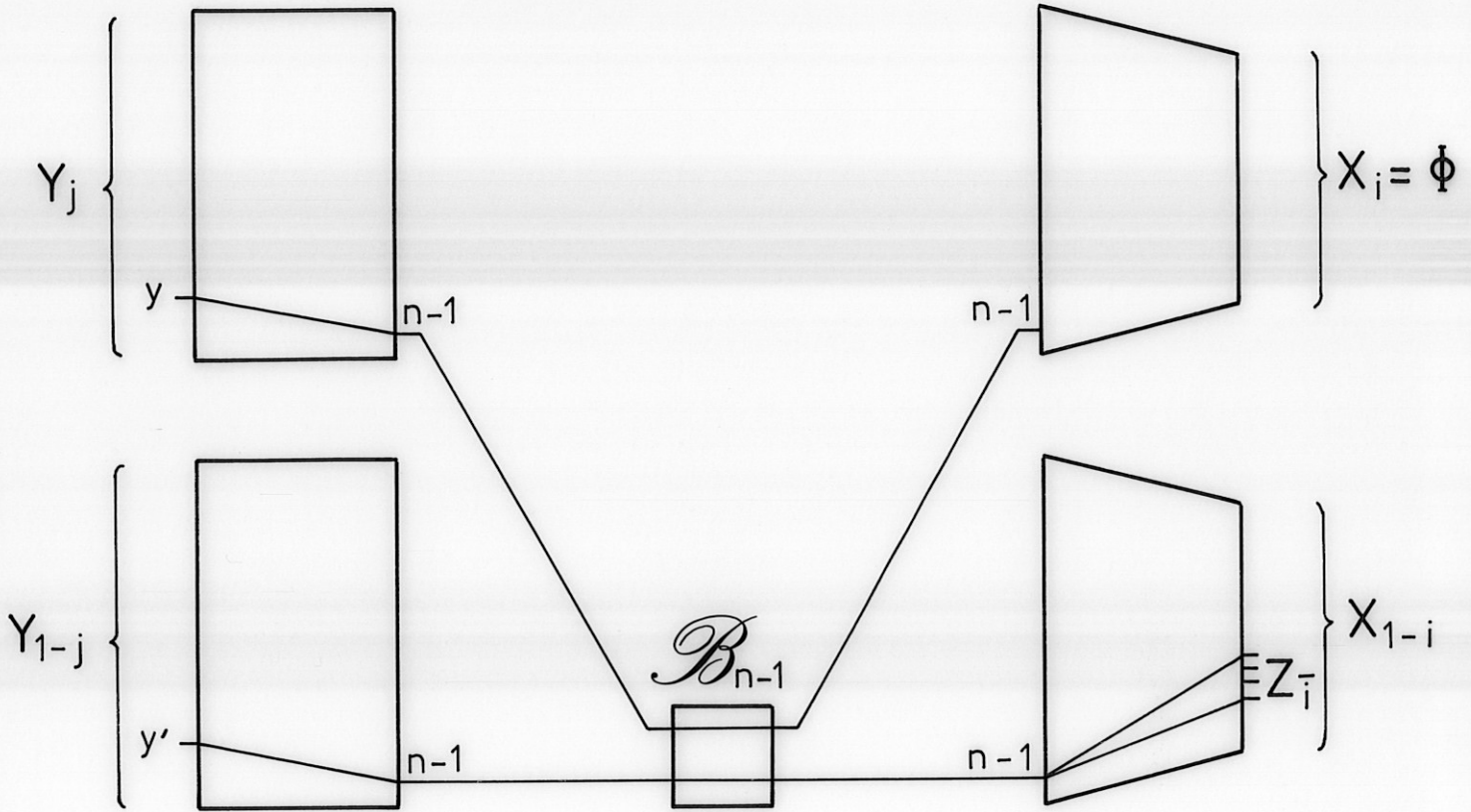
Figure 16.

Case 5. $X_i = \emptyset$.

$y \in Y_j$, $y' \in Y_{1-j}$.

$f^{-1}(y) = \emptyset$.

$f^{-1}(y') \neq \emptyset$.



$Z_i = \emptyset$.

$Z_{1-i} = f^{-1}(y')$

Delete y, y', Z_0, Z_1, B_{n-1} .

Figure 17.

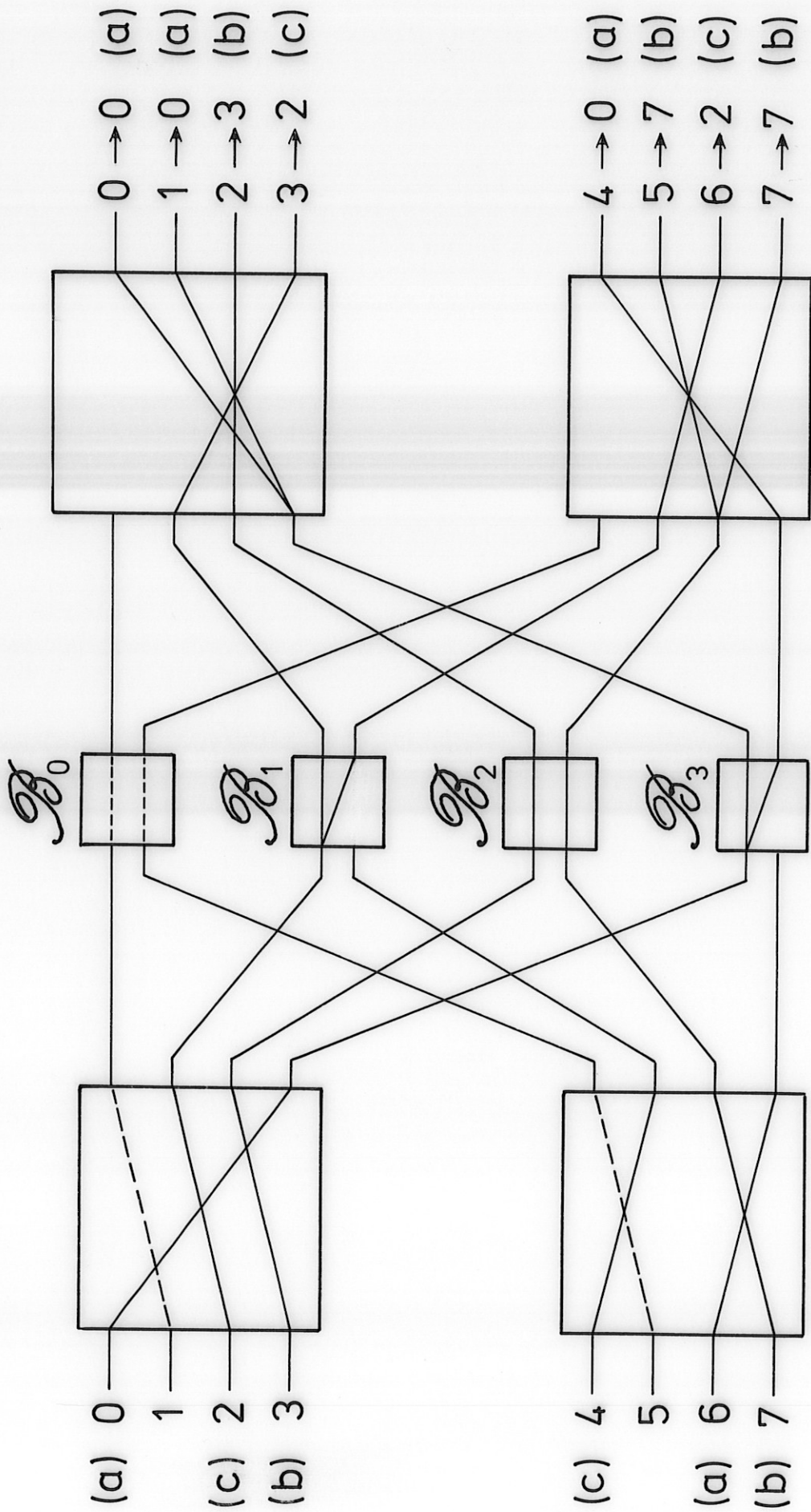


Figure 18.

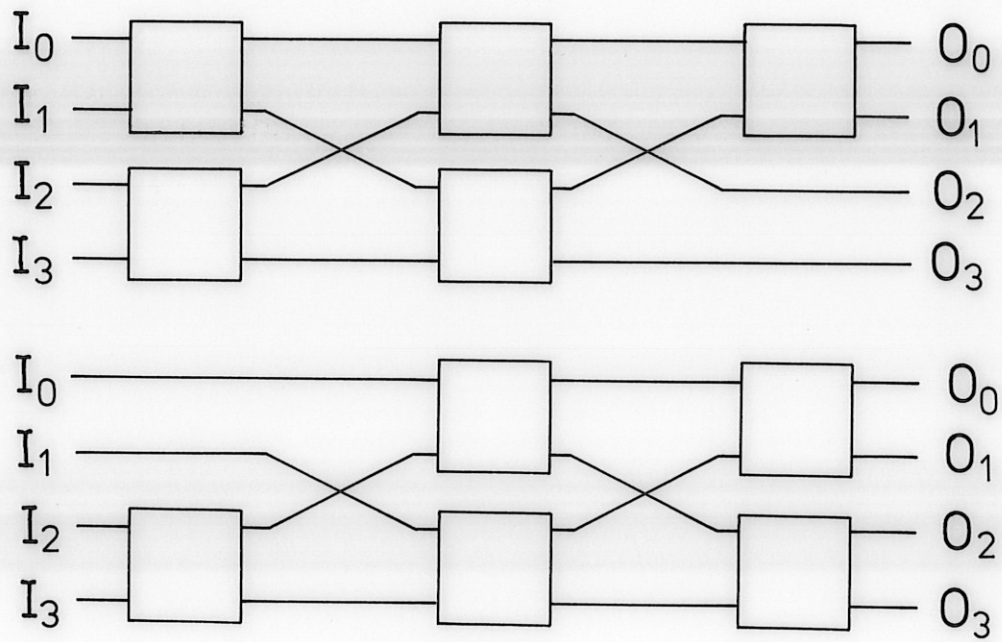


Figure 19.

Waksman's network.

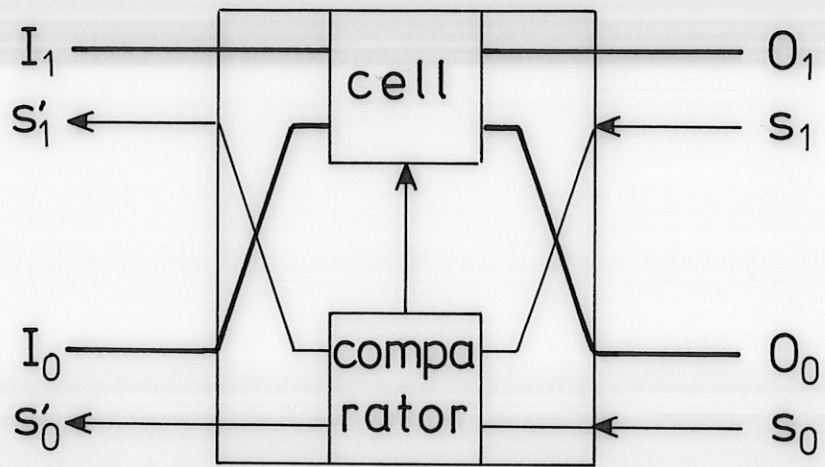
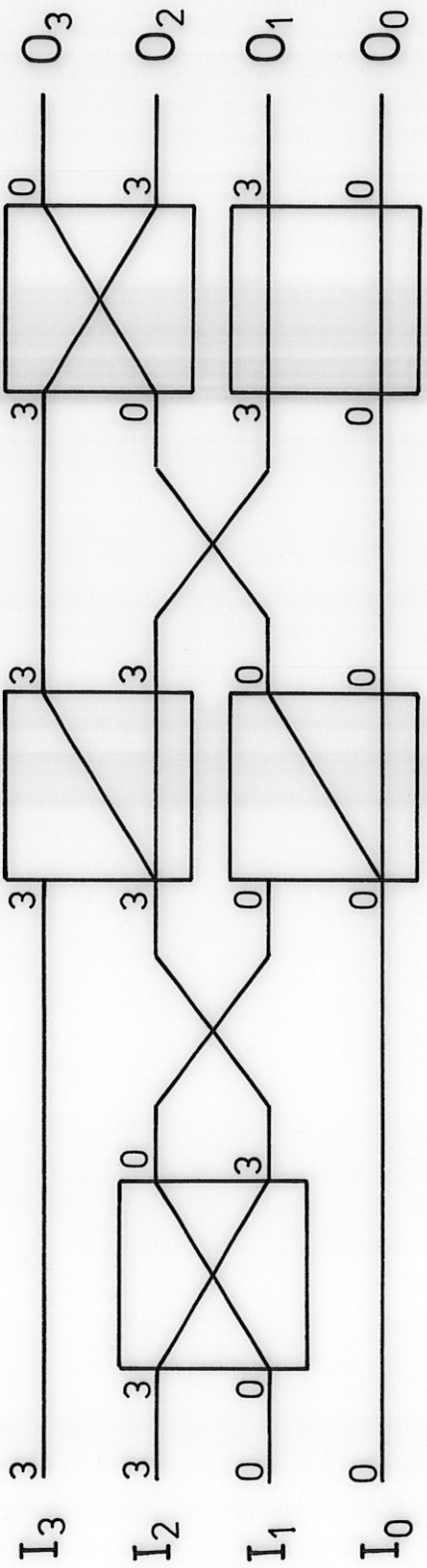


Figure 21.

Decentralized control.



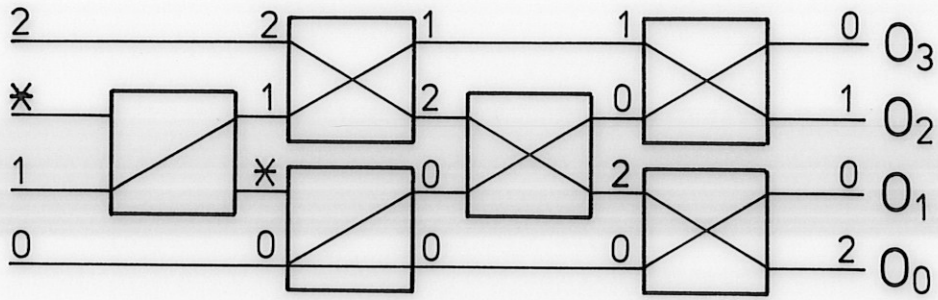
f : 0 → 0
 1 → 3
 2 → 3
 3 → 0

g : 0 → 0
 1 → 2
 2 → 2
 3 → 0

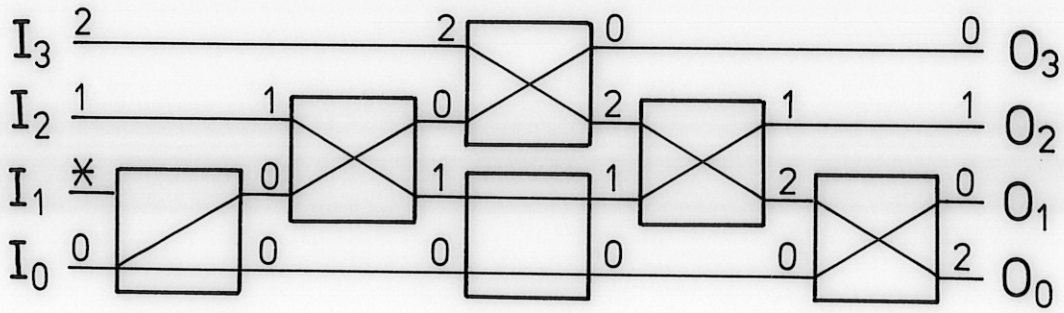
Figure 22.

$f : 0 \rightarrow 2$
 $1 \rightarrow 0$
 $2 \rightarrow 1$
 $3 \rightarrow 0$

(1)



(2)



(1) $f_1 : 0 \rightarrow 3$
 $1 \rightarrow 0$
 $2 \rightarrow 1$
 $3 \rightarrow 0$

(2) $f_2 : 0 \rightarrow 3$
 $1 \rightarrow 0$
 $2 \rightarrow 2$
 $3 \rightarrow 0$

Figure 23.

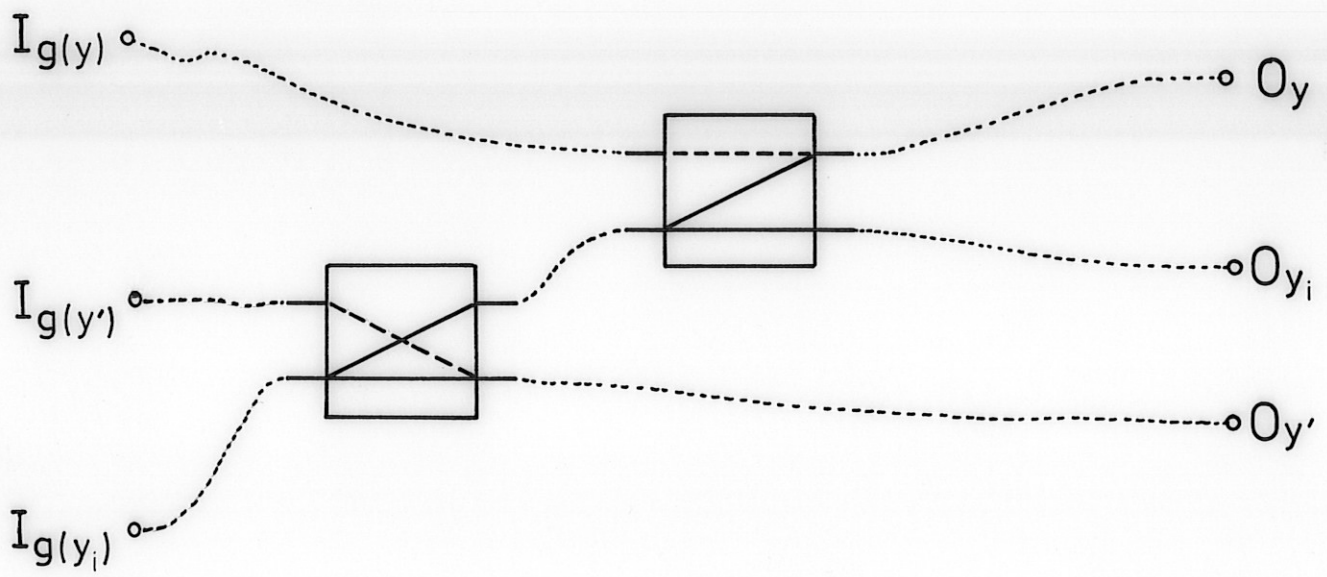


Figure 24. $y, y' \in F_i \setminus \{y_i\}$

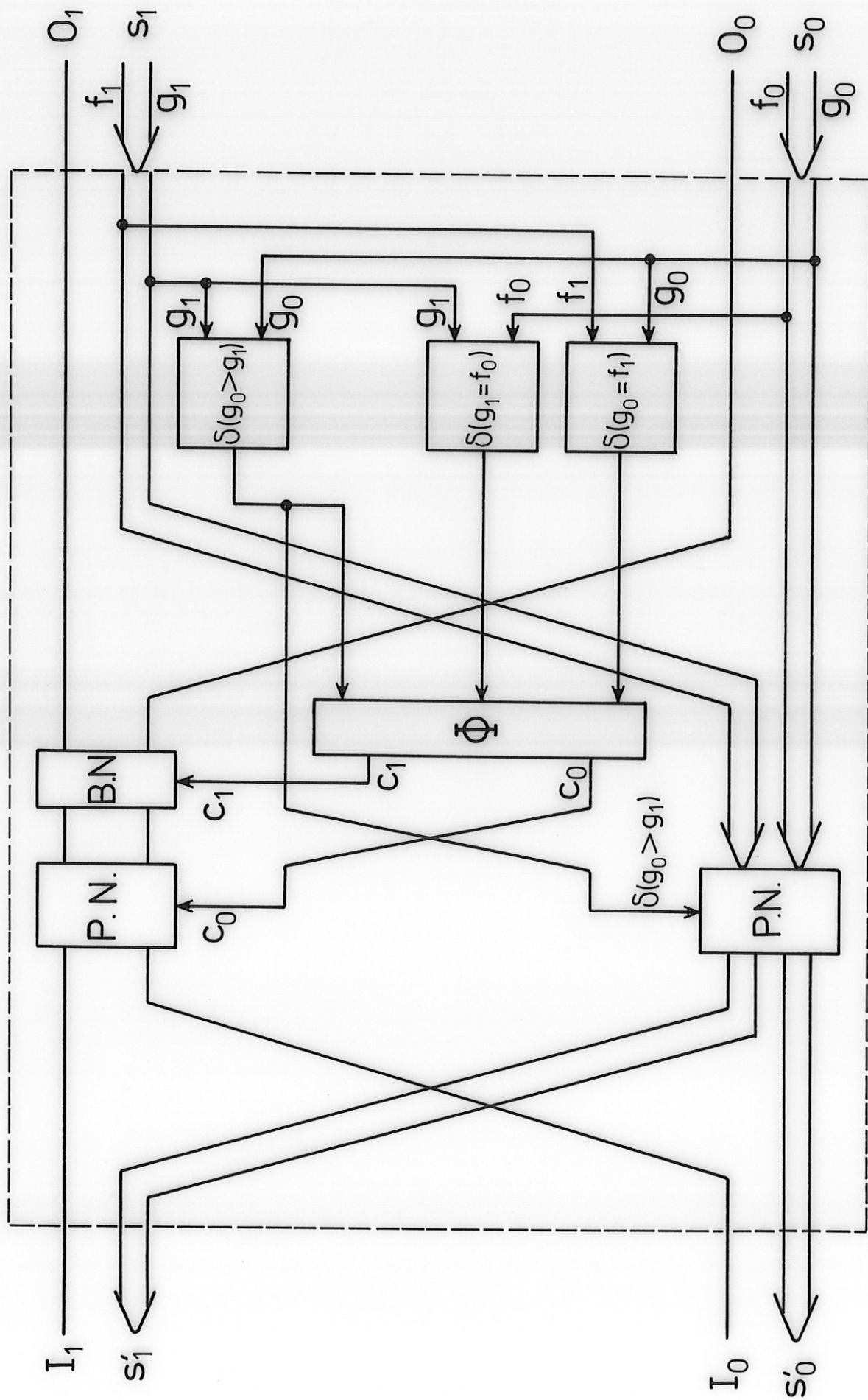


Figure 25.

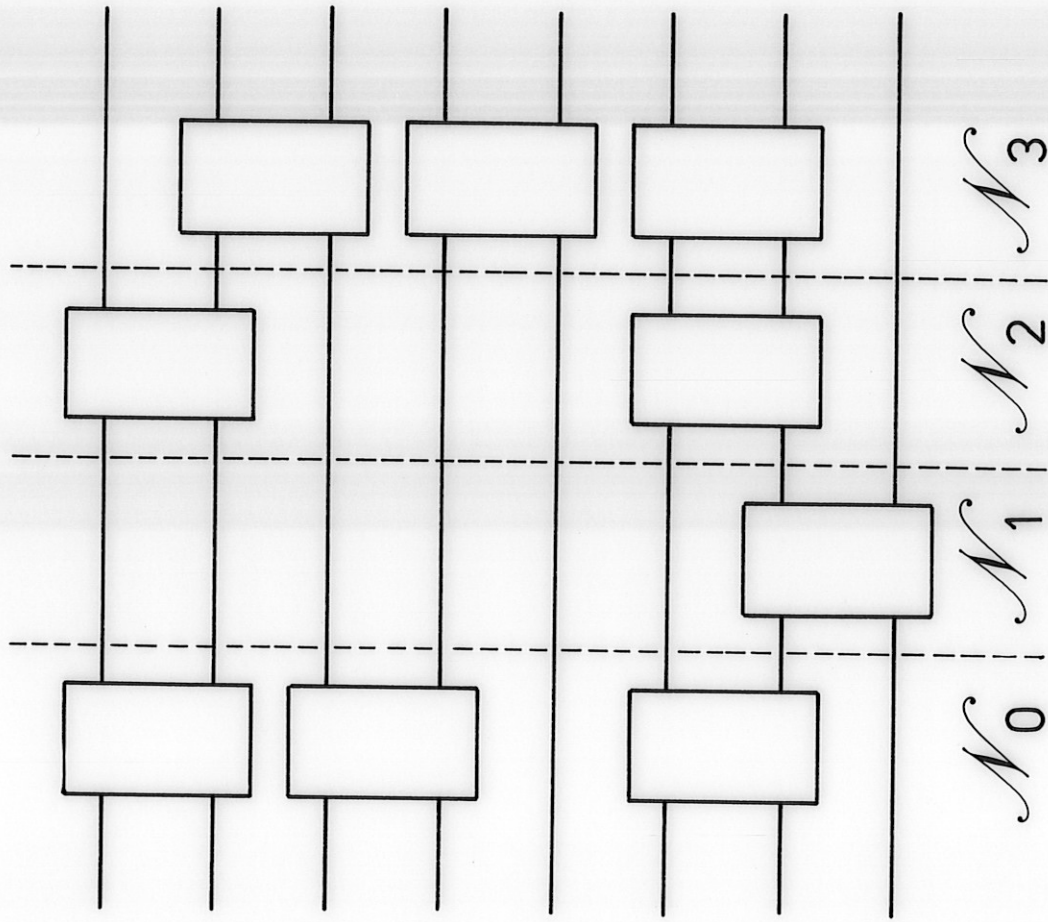
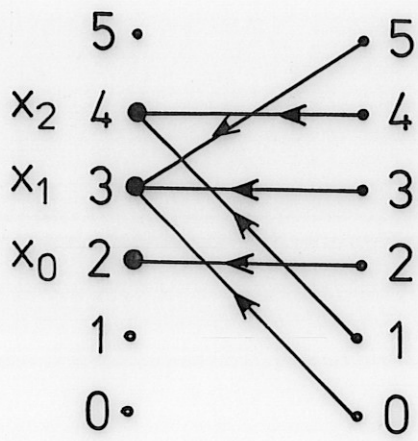
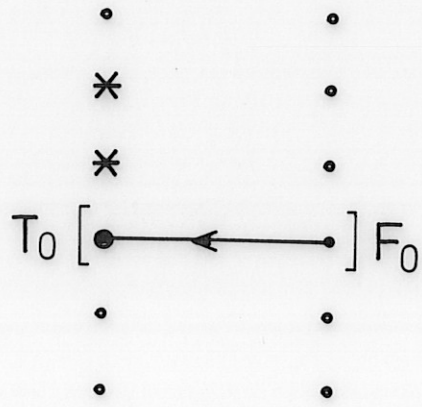


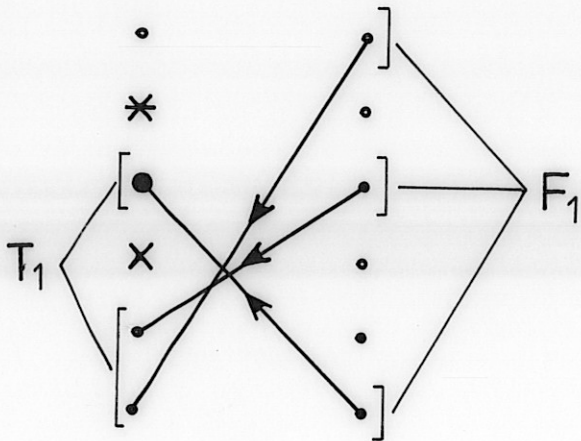
Figure 26. An adjacent network.



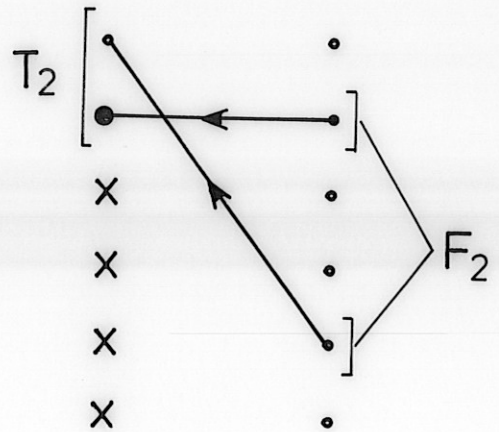
f



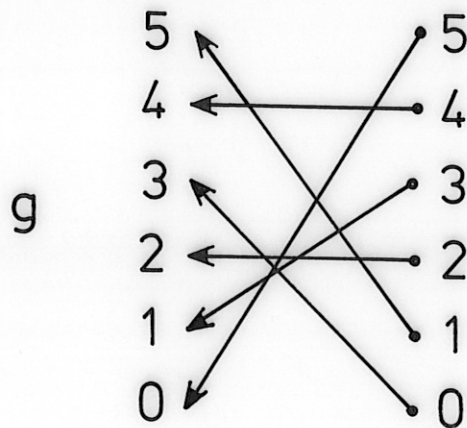
$g|_{F_0}$



$g|_{F_1}$



$g|_{F_2}$



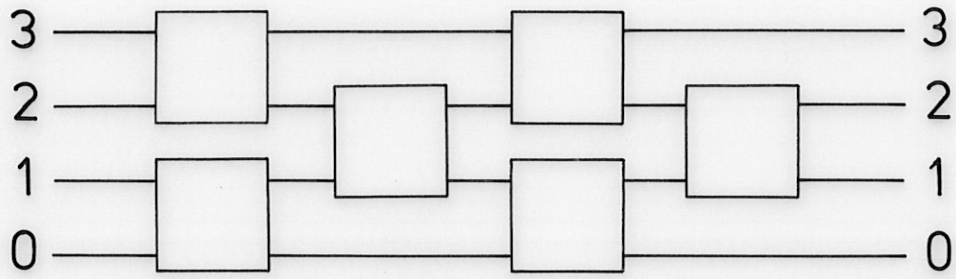
g

x = in $g(F_j)$ $j < i$

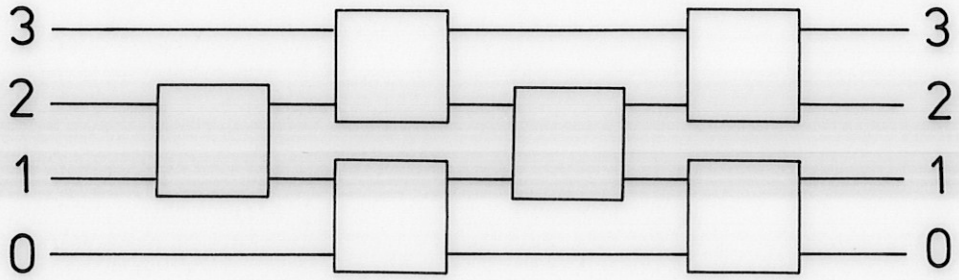
* = in $f(F_j)$ $j > i$

Figure 27.

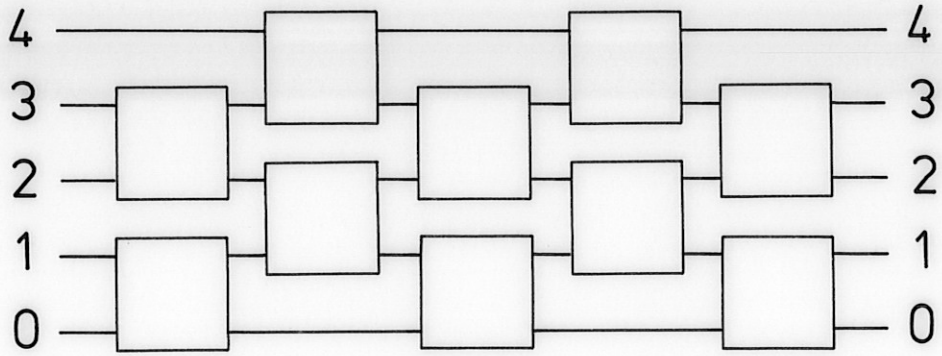
n = 4



or



n = 5



or

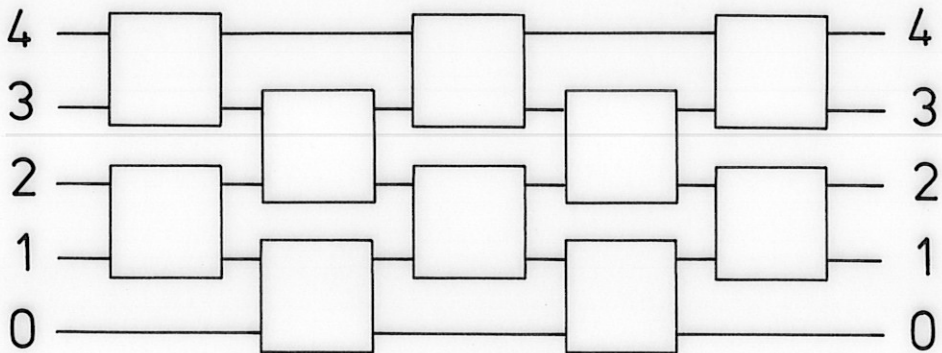


Figure 28.

The diamond array.

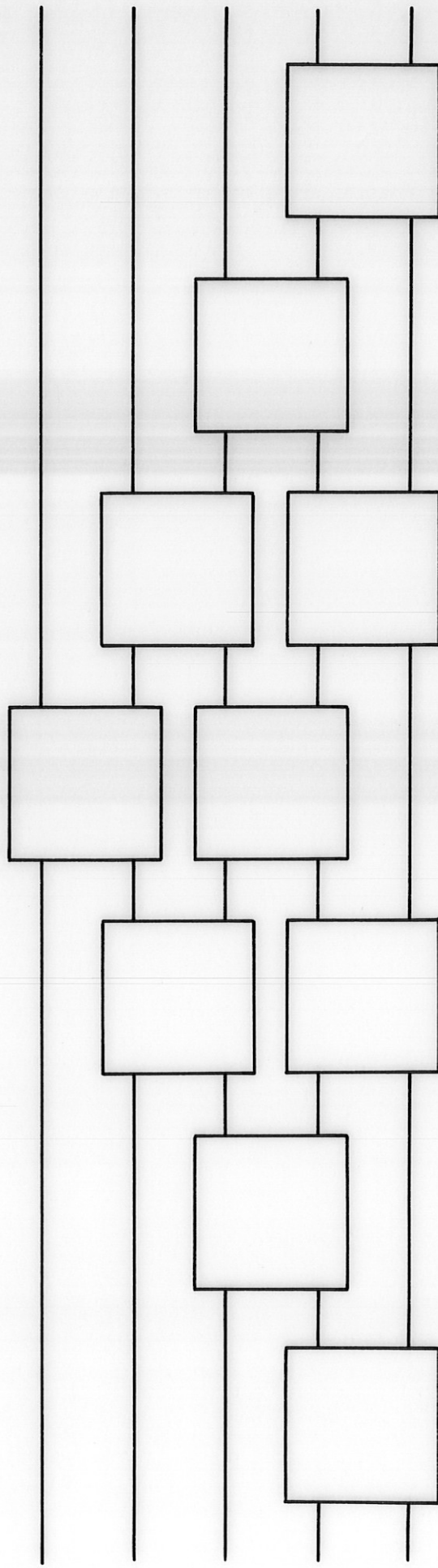
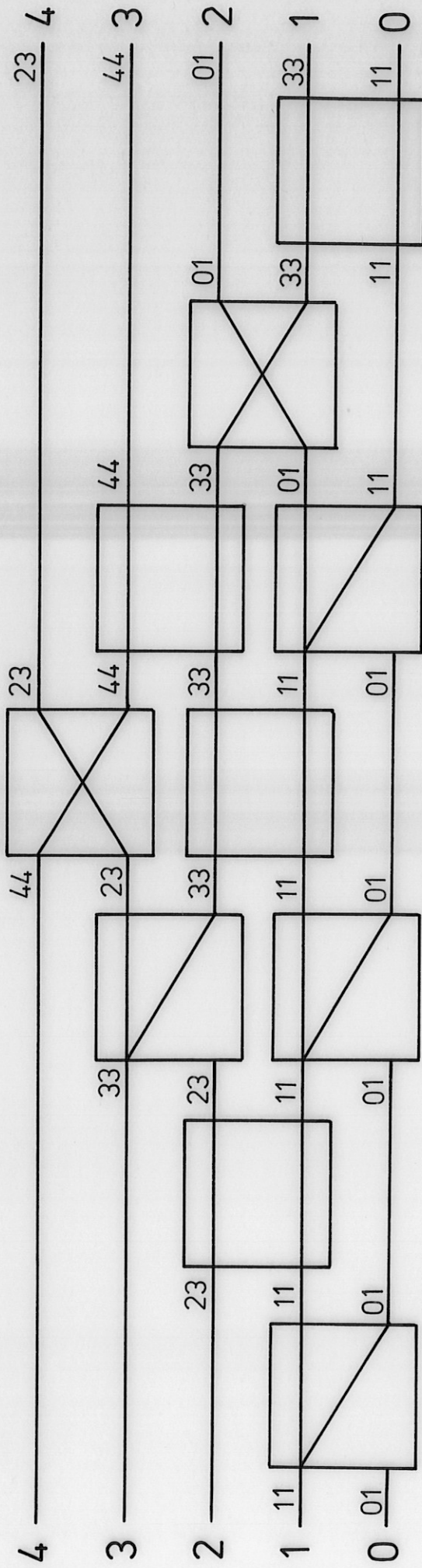


Figure 29. The triangular array ($n=5$)



f : 0 → 1
 1 → 3
 2 → 1
 3 → 4
 4 → 3

g : 0 → 1
 1 → 3
 2 → 0
 3 → 4
 4 → 2

Figure 30.