More on rearrangeability of combined $(2n-1)$-stage networks

Nabanita Das *

Advanced Computing and Microelectronics Unit, Indian Statistical Institute, 203, B.T. Road, Calcutta 700 035, India

Received 28 August 2002; received in revised form 17 July 2003; accepted 22 August 2004
Available online 30 December 2004

Abstract

This paper considers a class of combined $(2n-1)$-stage $N \times N$ interconnection networks composed of two $n(=\log_2 N)$-stage omega-equivalent networks $M(n)$ and $M'(n)$. The two networks are concatenated with the last stage of $M(n)$ overlapped with the first stage of $M'(n)$, forming a combined $(2n-1)$ stage network. Though both Benes network and $(2n-1)$-stage shuffle-exchange network belong to this class, the former one is a rearrangeable network, whereas the rearrangeability of the latter one is still an open problem. So far, there is no algorithm, in general, that may determine whether a given $(2n-1)$-stage combined network is rearrangeable or not. In this paper, a sufficient condition for rearrangeability of a combined $(2n-1)$-stage network has been formulated. An algorithm with time complexity $O(Nn)$ is presented to check it. If it is satisfied, a uniform routing algorithm with time complexity $O(Nn)$ is developed for the combined network. Finally, a novel technique is presented for concatenating two omega-equivalent networks, so that the rearrangeability of the combined network is guaranteed, and hence the basic difference between the topologies of a Benes network and a $(2n-1)$-stage shuffle-exchange network has been pointed out.

Keywords: Multistage interconnection network (MIN); Blocking MIN's; Rearrangeable networks; Topological equivalence; Omega-equivalent networks; Permutation routing

1. Introduction

For high-performance computing/communication applications, multistage interconnection networks (MIN's) have been studied extensively during the last two decades. With the advances in optical technology, optical multistage interconnection networks are also emerging as a promising networking choice [1] for connecting processors and/or memory modules. A full-access unique-path $N \times N$ multistage interconnection network [13,14], consisting of n stages of 2×2 switches, $(N = 2^n)$, is essentially a minimal structure that provides full-accessability with exactly one path between any input–output pair. But these networks are *blocking* by nature. Since, to route any arbitrary $N \times N$ permutation P through the

* Tel.: +91 33 2575 3006; fax: +91 33 2577 3035.
E-mail address: ndas@isical.ac.in

network, two input–output paths may require the same link, causing *conflict* which indicates that the paths can not exist simultaneously.

An $N \times N$ *rearrangeable* network is one which can connect its N inputs to its N outputs in all $N!$ possible ways, by rearranging the existing connections, if required. Hence, rearrangeability is a desirable feature of MIN's that can realize all permutations using minimal hardware. Benes network is a widely studied rearrangeable multistage architecture that uses the theoretically minimum number of stages required for rearrangeable operation [2,16]. This network provides an equal path length, low latency and low switch count. Its routing is also simple with a complexity $O(Nn)$ in general, though many important classes of permutations like BPC (bit permute complement), LC (linear-complement), etc. are found to be self-routable in it [3–5]. It can be used as optical multistage interconnection networks as well [7].

However, Benes network is essentially the concatenation of two *unique-path full-access blocking* MIN's, each with n stages, namely the baseline network $\beta(n)$ and the reverse-baseline network $\beta^{-1}(n)$ [13], overlapping the last stage of $\beta(n)$ with the first stage of $\beta^{-1}(n)$. Since then the study of the interconnection topologies of $(2n-1)$-stage networks formed by concatenating two *unique-path full-access* MIN's becomes a topic of high interest. In [6], the authors considered a class of combined $(2n-1)$ stage networks, represented as $\Delta \oplus \Delta'$, where both Δ and Δ' are omega-equivalent networks. They proposed an $O(N^4 n)$ algorithm to check the topological equivalence [8] of two such networks. In [9], Lee proved that $\Omega \oplus \Omega^{-1}$ is equivalent to Benes network, and, hence rearrangeable. But most interestingly, the rearrangeability of $\Omega \oplus \Omega$ still remains an open problem [17,18]. A coding scheme was proposed in [10] to check the equivalence of a limited class of $(2n-1)$-stage networks with Benes network. So far, there is no general algorithm to find whether any given $(2n-1)$-stage combined network is rearrangeable or not. Throughout this paper, the *equivalence* between networks means the *topological equivalence* [14].

The topological equivalence of any $(2n-1)$-stage network with Benes network proves that the former one is a rearrangeable network. But so far, it will need $O(N^4 n)$ time to decide the equivalence [6]. Moreover, still there is no result to show how we can correlate the information of topological equivalence with the exact routing algorithm of a network. In [11,12] some routing algorithms have been presented for symmetric $(2n-1)$ stage networks, like Benes and $\Omega \oplus \Omega^{-1}$ only.

In this paper, we focus on the problem of determining the rearrangeability of a combined $(2n-1)$-stage network $\Delta \oplus \Delta'$. Here, a sufficient condition for the rearrangeability of a combined network has been established, and an algorithm of $O(Nn)$ time complexity has been developed to check it. Next, given any such rearrangeable network, and an arbitrary $N \times N$ permutation P, a simple and uniform routing technique has been developed that routes P in $O(Nn)$ time. Finally, instead of overlapping the switches in the same physical position, an elegant rule has been proposed for concatenating two Ω-equivalent networks that guarantees the rearrangeability of the combined network, and hence points out the difference between the Benes network and the $\Omega \oplus \Omega$ networks.

The paper is organized as follows. In Section 2, some preliminary ideas have been introduced. In Section 3, the concatenation of two Ω-equivalent networks has been considered. In Section 4, the sufficient condition for rearrangeability of combined $(2n-1)$-stage networks has been proved. Section 5 describes the new concatenation technique to guarantee the rearrangeability. Section 6 presents some concluding remarks.

2. Preliminaries

So far, a large number of blocking MIN's have been proposed in the literature, e.g., baseline, omega, reverse-baseline, flip, etc. Here, we represent such an MIN with the following notations:

- inputs (and outputs) are labeled as: $0, 1, \ldots, N-1$, respectively, and each is represented uniquely by an n bit binary string $x_{n-1}x_{n-2}\ldots x_1x_0$;
- the stages are labeled as: $0, 1, \ldots, (n-1)$, from the input side towards the output side;

- the switches of each stage are labeled as: $0, 1, \ldots, (N/2 - 1)$.
- the output links of any switch with label j, $0 \leqslant j < N/2$, are marked as $2j$ (the upper link), and $(2j + 1)$ (the lower link), and each is represented uniquely by an n bit binary string $x_{n-1}x_{n-2}\ldots x_1x_0$;
- The path from an input link of a switch-j at any stage-i, $0 \leqslant i \leqslant (n-1)$ and $0 \leqslant j \leqslant (N/2 - 1)$, to its output is determined by the *routing bit* r_i. The path goes to the upper link $2j$, if $r_i = 0$, or to the lower link $(2j + 1)$, when $r_i = 1$. This bit r_i is termed as the *routing bit* for the path at stage i.

An 8×8 baseline network $\beta(3)$, with a labeling of the switches in *top-to-bottom* order is shown in Fig. 1. Also two paths $5 \rightarrow 2$ and $6 \rightarrow 3$ are shown which follow the routing bit sequences 010 and 011 respectively. It is to be noted that here the routing bit sequences are actually the corresponding destinations (2) and (3), respectively, in binary.

Here, the interconnection between stages i and $(i + 1)$, $0 \leqslant i \leqslant (n - 2)$, is represented as a unique permutation of $(n - 1)$ bits of any output link l of stage i, appended by the routing bit r_{i+1}, that maps l to link l' at the output of stage $(i + 1)$.

For stage-0, the mapping of original inputs to the outputs of stage 0 is considered.

It is to be noted that the labels here indicate some logical names to identify the switches/links uniquely, which help to formulate the *topology describing rules* in the form of permutation of bits described below.

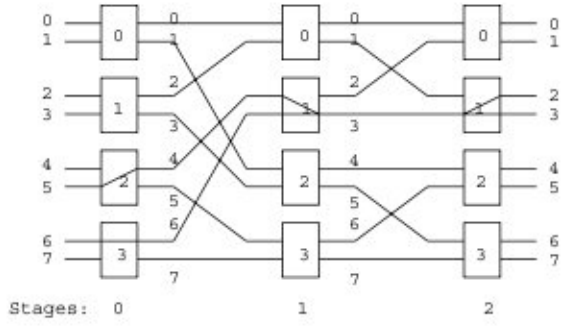

Fig. 1. An 8×8 baseline network $\beta(3)$ with two paths $5 \rightarrow 2$ and $6 \rightarrow 3$.

Example 1. For $\beta(3)$ with the labeling of the switches shown in Fig. 1, the interconnection between stages 0 and 1 can be represented as a mapping $f_1[x_2x_1x_0] \rightarrow x_0x_2r_1$, i.e., any output link of stage 0, represented as $x_2x_1x_0$, is connected to a link $x_0x_2r_1$, at the output of stage 1, where r_1 is the routing bit for the path at stage-1. Note that the permutation f_1 is invariant for all the links at the output of stage 0.

Definition 1. Given an MIN $M(n)$, with a labeling of the links, if there exists a mapping $f_i[x_{n-1}x_{n-2}\ldots x_1x_0] \rightarrow y_{n-1}y_{n-2}\ldots y_1r_i$, for each stage i, $0 \leqslant i \leqslant (n-1)$, where, $y_{n-1}y_{n-2}\ldots y_1$ is a permutation of any $(n-1)$ bits of $(x_{n-1}, x_{n-2}, \ldots, x_1, x_0)$, and r_i is the routing bit, such that any link $l_i{:}x_{n-1}x_{n-2}\ldots x_1x_0$ at the output of stage $(i-1)$ is connected to link l'_{i+1} at the output of stage i, represented as $f_i[l_i] \rightarrow l'_{i+1}$, where $l'_{i+1} : y_{n-1}y_{n-2}\cdots y_1r_i$, the labeling of the switches is termed as a basic labeling, and f_i's are defined as the *i-mappings* for $M(n)$.

For simplicity, we represent an *i-mapping* as $f_i \rightarrow y_{n-1}y_{n-2}\ldots y_1r_i$, assuming that it is always applied on $[x_{n-1}x_{n-2}\ldots x_1x_0]$.

Example 2. In Fig. 1, with the labeling of the links shown, the *i-mappings* for the baseline network $\beta(3)$ are given by

$f_0 \rightarrow x_2x_1r_0$, $f_1 \rightarrow x_0x_2r_1$, and $f_2 \rightarrow x_2x_0r_2$, where, r_i is the routing bit at stage-i. Since with the labeling of $\beta(3)$ shown in Fig. 1, the *i-mappings* exist for all i, $0 \leqslant i \leqslant 2$, it is a *basic labeling*.

In general, for $\beta(n)$, the *i-mappings* are

$f_0 \rightarrow x_{n-1}\ldots x_1r_0$, $f_1 \rightarrow x_0x_{n-1}\ldots x_2r_1$, $f_2 \rightarrow x_{n-1}x_0x_{n-2}\ldots x_2r_2$, $\ldots$, and $f_{n-1} \rightarrow x_{n-1}x_{n-2}\ldots x_2x_0r_{n-1}$.

Remark 1. Given an MIN $M(n)$ the basic labeling is not unique.

Definition 2. Given an MIN $M(n)$, the labeling of all the inputs, and the switches of each stage i, $0 \leqslant i \leqslant (n-1)$, in top-to-bottom order (as shown for $\beta(3)$ in Fig. 1) is termed as top-to-bottom labeling.

Remark 2. For most of the well-known full-access unique-path MIN's $M(n)$, e.g., baseline, reverse-baseline, omega, inverse-omega, flip, cube etc. [13], the top-to-bottom labeling is found to be a basic labeling.

Example 3. Fig. 2 shows an 8×8 omega network $\Omega(3)$ with top-to-bottom labeling.

It is to be noted that the *i-mappings* for $\Omega(3)$ are given as

$f_0 \rightarrow x_1 x_0 r_0$, $f_1 \rightarrow :x_1 x_0 r_1$, $f_2 \rightarrow :x_1 x_0 r_2$, where, r_i is the routing bit at stage-i, $0 \leqslant i \leqslant 2$.

In general, for $\Omega(n)$, $f_i \rightarrow :x_{n-2} x_{n-3} \ldots x_0 r_i$ for $0 \leqslant i \leqslant (n-1)$.

Given an MIN $M(n)$ with a basic labeling of its links, any input–output *path* through $M(n)$ starts from an input, selects either the upper link, or the lower link of a switch, determined by the routing bit r_i, traverses via consecutive stages i, $0 \leqslant i \leqslant (n-1)$, and reaches the final output. Therefore, we may represent a path as a sequence of links, starting from the input x $(=x_{n-1}x_{n-2}\ldots x_1 x_0)$, and following the output link l_{i+1} at stage i, given by $f_i[l_i]$, finally reaching the output y $(=y_{n-1}y_{n-2}\ldots y_1 y_0)$.

Example 4. Given the labeling of the baseline network $\beta(3)$, as shown in Fig. 1, and corresponding *i-mappings* $f_0 \rightarrow x_2 x_1 r_0$, $f_1 \rightarrow :x_0 x_2 r_1$, and $f_2 \rightarrow :x_2 x_0 r_2$, (Example 2), the sequence of links followed by the input–output path $5 \rightarrow 2$ are given by

Input $5 : 101 \xrightarrow{f_0} 10r_0 \xrightarrow{f_1} r_0 1 r_1 \xrightarrow{f_3} r_0 r_1 r_2$.

Now, since the network follows destination tag routing, and the destination is the final output 2 i.e. 010 in binary, $r_0 = 0$, $r_1 = 1$, $r_2 = 0$.

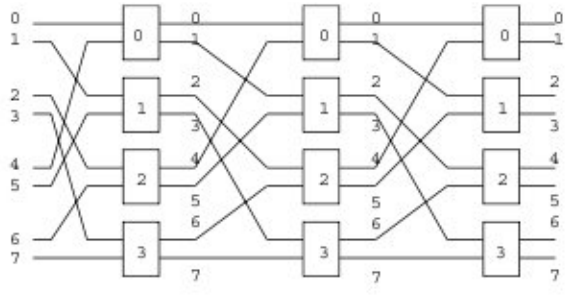

Fig. 2. An 8×8 omega network $\Omega(3)$ with top-to-bottom labeling.

Substituting these values of r_i's on each link, we get the path as the sequence $5 \rightarrow 4 \rightarrow 3 \rightarrow 2$, shown in Fig. 1.

Now, to set up paths from input to output on an MIN, there would be a conflict at a stage-i, if and only if two or more paths need the same link at the output of any stage i.

Example 5. Fig. 1 shows that in $\beta(3)$, the path $5 \rightarrow 2$ conflicts with the path $6 \rightarrow 3$ in stage 1, represented as:

$5 \rightarrow 4 \rightarrow 3 \rightarrow 2$, and $6 \rightarrow 6 \rightarrow 3 \rightarrow 3$.

Both the paths require the same link 3 at the output of stage 1, and therefore conflict.

The concept of *topological equivalence* of two MIN's was first introduced in [14,8], and has been studied extensively thereafter. Given an MIN M with k stages, its *topology graph* $TG(M)$, is a graph where each switch is represented as a distinct vertex, and the links between switches are represented by edges between corresponding vertices. Obviously, the topology graph would be a level graph, with $N/2$ vertices at each level-i, referred as $V_i(M)$ representing the $N/2$ switches in stage i, $0 \leqslant i \leqslant k$. In [6], the topological equivalence has been defined in the following way:

Definition 3. Any two k-stage $N \times N$ MIN's M_1 and M_2 are called topologically equivalent if and only if there is an isomorphic mapping ψ from $TG(M_1)$ to $TG(M_2)$, such that $\psi(v) \in V_i(M_2)$, $\forall v \in V_i(M_1)$ $i = 0, 1, \ldots, k-1$.

Given any n-stage $N \times N$ MIN M, by the *Ω-equivalence checking algorithm* presented in [6], we may check its equivalence with $\Omega(n)$ network in $O(Nn)$ time. Example 3, shows a basic labeling for $\Omega(3)$ network, that results a set of *i-mappings* describing the interconnection topology of the network.

Lemma 1. *For any Ω-equivalent MIN $M(n)$ there will exist at least one basic labeling, resulting a set of i-mappings, $\forall i$, $0 \leqslant i \leqslant (n-1)$.*

Proof. Let the network $\Omega(n)$ is given with all its inputs, switches and output links labeled according to *top-to-bottom* labeling. It has already been shown that it is a basic labeling for $\Omega(n)$.

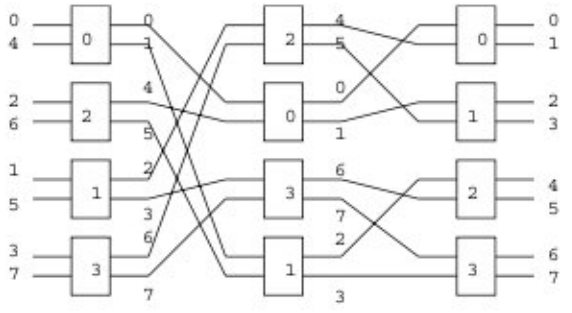

Fig. 3. An 8×8 Ω-equivalent network $M(3)$.

From the definition of equivalence, the topology graph of $M(n)$ is isomorphic with that of $\Omega(n)$ network. Therefore, the nodes of each stage i of $M(n)$ have a one-to-one correspondence with those of stage i of $\Omega(n)$, for $0 \leqslant i \leqslant (n-1)$. Therefore, we can re-label the network inputs and the switches of each stage i of $M(n)$ according to the labels of those of stage i in $\Omega(n)$. Now it is evident that this labeling of the nodes of $M(n)$ will result the set of *i-mappings*, same as those of $\Omega(n)$, as given in Example 3. Therefore, this labeling of switches of $M(n)$ comprises a basic labeling for $M(n)$. Hence the proof. □

Example 6. Fig. 3 shows an 8×8 MIN $M(3)$ (equivalent to Ω network) with a labeling according to its topology isomorphism with $\Omega(3)$.

The corresponding *i-mappings* are given below.

f_0: $x_1 x_0 r_0$, f_1: $x_1 x_0 r_1$, f_2: $x_1 x_0 r_2$. Since f_i exists for $\forall i$, $0 \leqslant i \leqslant 2$, the labeling shown is a basic labeling of $M(3)$. Note that the *i-mappings* are same as those of $\Omega(3)$.

3. Concatenation of Ω-equivalent MIN's

Given any n-stage $N \times N$ MIN $M(n)$, by the *Ω-equivalence checking algorithm* in [6], we may check whether it is equivalent to $\Omega(n)$ network. If yes, we can also compute a basic labeling of the links of M and the corresponding *i-mappings* of $M(n)$ by the algorithms presented in Section 4.

Next, two Ω-equivalent networks, each with a basic labeling, are concatenated to form a *combined* $(2n-1)$-stage network $\Delta \oplus \Delta'$. So far, *concatenation* meant the overlapping of each switch of the last stage of Δ with that at the same physical position in the first stage of Δ'. But it does not necessarily retain the topological equivalence of the combined network, so formed.

Definition 4. Given two Ω-equivalent networks Δ and Δ', each with a basic labeling, in the *combined* network $\Delta \oplus \Delta'$, the labels of the network inputs and the output links of stages i, $0 \leqslant i \leqslant (n-2)$, are kept same as they were in Δ, and for stages i, $(n-1) \leqslant i \leqslant (2n-2)$, same as those in Δ'. This overlapping actually causes a one-to-one mapping of the final output links of Δ to the links at the output of stage-0 of Δ'. This mapping is defined as the concatenation mapping.

Example 7. Fig. 4 shows the *combined* $(2n-1)$-stage network $\Omega \oplus M$, where M is the MIN shown in Fig. 3. In stage-2, the corresponding switch labels x in $\Omega(3)$ has been shown as (x). It is to be noted that the concatenation mapping is given by

$$C : \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 0 & 1 & 4 & 5 & 2 & 3 & 6 & 7 \end{pmatrix}$$

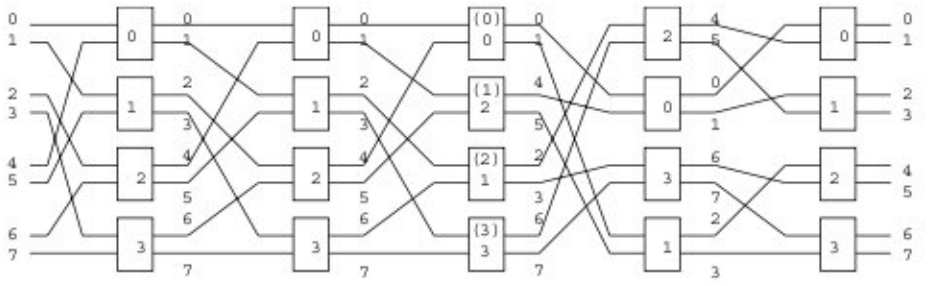

Fig. 4. The combined 8×8 network $\Omega \oplus M$.

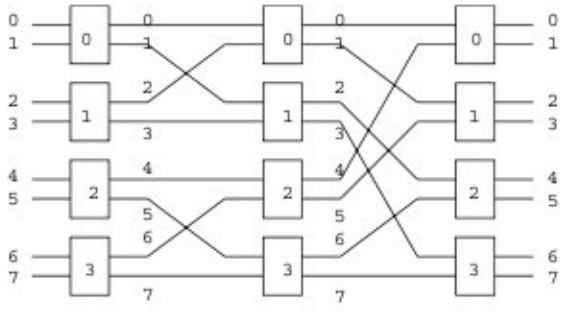

Fig. 5. An 8×8 reverse baseline network $\beta^{-1}(3)$.

Example 8. Fig. 6 shows a Benes network $\beta(3)\oplus\beta^{-1}(3)$, a concatenation of a baseline network $\beta(3)$ and a reverse-baseline network $\beta^{-1}(3)$, shown in Figs. 1 and 5, respectively. Here the concatenation mapping C is an identity permutation.

Remark 3. In [12] Feng et al. considered the networks, where the outside-in coding of the switches of the center stage of a $(2n-1)$-stage network are identical from both sides. It corresponds to the case where the concatenation mapping C is always an identity permutation. But, in this paper, a more general class of combined networks has been considered.

Lemma 2. *Given any combined $(2n-1)$-stage network $\Delta\oplus\Delta'$, where both Δ and Δ' are equivalent to Ω network, there exist i-mappings for all i, $0\leqslant i\leqslant(2n-2)$, if and only if the concatenation mapping is a bit-permute permutation.*

Proof. Let us consider two Ω equivalent networks Δ and Δ' with the set of *i-mappings* $\{f_0,f_1,\ldots,f_{n-1}\}$, and $\{f'_0,f'_1,\ldots,f'_{n-1}\}$, respectively. These two are combined to form the network $\Delta\oplus\Delta'$.

If part. Let us first assume that the concatenation mapping C is a bit-permute permutation. Then obviously, with the labeling described in Definition 4, for the combined network $\Delta\oplus\Delta'$, the *i-mappings* are given by $F_i=f_i$, for $0\leqslant i\leqslant(n-2)$, $F_i=f'_i$, for $n\leqslant i\leqslant(2n-2)$, and $F_i=C\circ f_i$ for $i=(n-1)$, where $C\circ f_i$ is the composition of two bit-permute permutations, C and f_i, and hence another bit-permute permutation. Therefore, it is evident that for $\Delta\oplus\Delta'$, the *i-mappings* exist for all i.

Only if part. Let us assume that for $\Delta\oplus\Delta'$, the *i-mappings* exist for all i. Hence for $i=(n-1)$, the mapping rule from the output links of stage $(n-2)$ to the output links of stage $(n-1)$ follows a bit-permute rule. It will be true only if the concatenation mapping C itself is also a bit-permute permutation. Hence the proof. □

Example 9. For the network $\Omega\oplus M$ shown in Fig. 4, the concatenation mapping C is a bit-permute permutation given by: C: $x_1x_2x_0$ (Example 7).

The *i-mappings* for the network are given below.

F_0: $x_1x_0r_0$, F_1: $x_1x_0r_1$ (same as those of Ω network),
F_2: $C\circ(x_1x_0r_2)$: $x_0x_1r_2$, and
F_3: $x_1x_0r_3$, F_4: $x_1x_0r_4$ (same as the *i-mappings* of last two stages of $M(3)$ as given in Example 6).

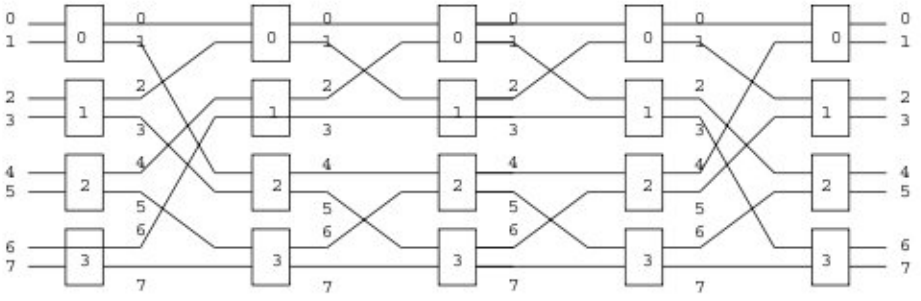

Fig. 6. An 8×8 Benes network $\beta(3)\oplus\beta^{-1}(3)$.

Remark 4. For any network $\Delta \oplus \Delta'$, where the *i-mappings* for Δ are f_i, and for Δ' are f'_i for $0 \leqslant i \leqslant (n-1)$, and if the concatenation mapping be the identity permutation, the *i-mappings* for the combined network will be F_i, where $F_i = f_i$, for $0 \leqslant i \leqslant (n-1)$, and $F_{n-1+j} = f'_j$, for $1 \leqslant j \leqslant (n-1)$.

Example 10. For the network $\Omega(3) \oplus \Omega(3)$, it is to be noted that the concatenation mapping is an identity permutation. Hence, *i-mappings* are given as

F_i: $x_1 x_0 r_i \; \forall i, 0 \leqslant i \leqslant (2n-2)$.

4. Rearrangeability of combined networks

An $N \times N$ network is called to be rearrangeable, if all $N \times N$ permutations are conflict-free on it. Now Benes network is a rearrangeable one, whereas the rearrangeability of $\Omega \oplus \Omega$ is still an open problem, though both of them are constructed by combining two Ω-equivalent networks. So, the question is that in the combined $(2n-1)$-stage networks, what factors actually determine the rearrangeability of the network?

It is to be noted that in a combined $(2n-1)$-stage network $\Delta \oplus \Delta'$, two Ω-equivalent networks are concatenated by superimposing a switch of the last stage of Δ to that in the first stage of Δ', which are in the same physical position. They may be logically different according to the relation of topology isomorphism. Hence, first of all, the *i-mappings* may or may not exist for the combined $(2n-1)$-stage network. Also, even if it exists, it may vary from network to network depending on the physical positions of the corresponding switches in Δ and Δ'. The most important point is that, even if the concatenation mapping remains the same the behavior of the networks regarding rearrangeability may differ. As it is evident from the networks $\Omega \oplus \Omega^{-1}$, and $\Omega \oplus \Omega$. Given the *i-mappings* of a combined $(2n-1)$-stage network, here a sufficient condition is formulated for rearrangeability of a combined network, in general.

Definition 5. Given a combined $(2n-1)$-stage network $\Delta \oplus \Delta'$, for a particular input-output path, the *routing bits* for stages i, $(n-1) \leqslant i \leqslant (2n-2)$ are predetermined by the destination tag, but the *routing bits* r_i, $0 \leqslant i \leqslant (n-2)$ are arbitrary, and are referred here as arbitrary routing bits (AR-bits).

4.1. Windows and rearrangeability

Here the notion of windows [15] has been utilized to represent a path through the network, and to correlate the characteristics of windows with the rearrangeability of a combined network.

Definition 6. For a given combined $(2n-1)$-stage network $\Delta \oplus \Delta'$, with a set of *i-mappings*, $\{F_0, F_1, \ldots, F_{2n-2}\}$, an input–output path $x \to y$, can be represented as a sequence of links $x \to l_1 \to l_2 \to \cdots l_{2n-2} \to y$, where, l_k is the link (in binary), the path follows at the output of stage $(k-1)$, $1 \leqslant k \leqslant (2n-2)$, and is given by $l_k = F_{k-1}[l_{k-1}]$, l_0 is the input x, and l_{2n-1} is the output y. Now, given an $N \times N$ permutation, at any stage k, the set of links followed by individual paths is represented as an $N \times n$ matrix, called window W_k, where each row L_j, $0 \leqslant j \leqslant (N-1)$ of window W_k represents the link in binary), followed by the path from input j, at the output of stage k.

Remark 5. A permutation is conflict-free on a combined $(2n-1)$-stage network, if and only if there exist windows W_k $\forall k, 0 \leqslant k \leqslant (2n-1)$, such that all rows of each window are distinct.

It is to be noted that window W_0 and window $W_{(2n-1)}$ are, respectively the $N \times n$ matrices of inputs and outputs, i.e., for any $N \times N$ permutation all rows of W_0 and $W_{(2n-1)}$ are distinct.

Definition 7. For a combined $(2n-1)$-stage network, $\Delta \oplus \Delta'$, if *i-mappings* F_i exist for all i, $0 \leqslant i \leqslant (2n-2)$, each window W_k, $0 \leqslant k \leqslant (2n-1)$ can be represented uniquely as a string $S_k = F_{k-1}[S_{k-1}]$, where S_0 is the input string, i.e., $S_0 = x_{n-1} x_{n-2} \ldots x_1 x_0$. S_k is defined as the characteristic string of W_k.

Example 11. For an 8×8 Benes network $\beta(3) \oplus \beta^{-1}(3)$, the characteristic strings for the windows are given by

S_0: $x_2x_1x_0$, S_1: $F_0[S_0] \rightarrow x_2x_1r_0$, S_2: $F_1[S_1] \rightarrow r_0x_2r_1$, S_3: $F_2[S_2] \rightarrow r_0r_1r_2$, S_4: $F_3[S_3] \rightarrow r_0r_2r_3$, and S_5: $F_4[S_4] \rightarrow r_2r_3r_4$, respectively.

Definition 8. From a conflict-free window W_k, if one column is deleted, the pair of rows having the same bit pattern are called conjugate rows.

Theorem 1. *In a combined $(2n-1)$-stage network, $\Delta \oplus \Delta'$, if i-mappings exist for all i, $0 \leqslant i \leqslant (2n-2)$, and each AR-bit r_j, $0 \leqslant j \leqslant (n-2)$, occurs only in each S_k, for $(j+1) \leqslant k \leqslant (2n-2-j)$, the network is rearrangeable.*

Proof. In a combined $(2n-1)$-stage network $\Delta \oplus \Delta'$, let us assume that all the *i-mappings* exist for $0 \leqslant i \leqslant (2n-2)$. At stage-0, the characteristic string S_0 is the input bit string $x_{n-1}x_{n-2} \ldots x_1x_0$, and also the string S_{2n-1} is the output bit string $y_{n-1}y_{n-2} \ldots y_1y_0$. Now from S_0, S_1 is obtained by applying the *i-mapping* F_0, i.e., deleting one bit x_i and adding the routing bit r_0, and permuting the bits according to F_0. In general, any S_i is obtained by permuting $(n-1)$ bits of S_{i-1} according to F_{i-1}, and adding the routing bit r_{i-1}.

It is to be noted that the routing bits for the last n stages, namely r_{n-1}, r_n, $\ldots$, r_{2n-2} are the output bits $y_{n-1}, y_{n-2}, \ldots, y_1$, y_0, respectively. They are already fixed by the given permutation P. The remaining routing bits, namely, r_0, r_1, $\ldots$, r_{n-2} are the AR-bits which are to be determined to make all the paths of P conflict-free, i.e., to make all the rows in each window W_k, $0 \leqslant k \leqslant (2n-1)$, distinct.

However, for any permutation P, W_0 is always conflict-free. Same is true for W_{2n-1}.

Let it be assumed that for the given network, the characteristic strings are such that any *AR-bit* r_j, $0 \leqslant j \leqslant (n-2)$, occurs only in each S_k, $(j+1) \leqslant k \leqslant (2n-2-j)$. Hence, the first routing bit r_0 which is an AR-bit appearing first in window W_1, has to occur in each window W_k, for $1 \leqslant k \leqslant (2n-2)$.

Since W_0 is conflict-free, and W_1 comprises of the same $(n-1)$ columns of W_0 with an additional column for r_0, it would be conflict-free, if and only if complementary bits are assigned for the two conjugate rows of W_0. The same will be true for window W_{2n-2} also. Let us start from any arbitrary row-j of W_1 with all bits fixed from W_0 except the bit r_0. Let us put $r_0 = 0$, say, in that row. Now let us find the *conjugate row* j', and put $r_0 = 1$, so that the two rows become distinct and hence conflict-free in W_1.

Next, in W_{2n-2}, j' the conjugate row of j' is found, with all bits fixed from W_{2n-1} except the bit r_0. r_0 is assigned as $r_0 = 0$, so that rows-j' and j'' become distinct and hence conflict-free in W_{2n-2}. Next the conjugate row of j'' is examined in W_1, and is assigned as $r_0 = 1$. This process is repeated unless a cycle is complete. Next it is to be started arbitrarily again from any row left in W_1, and the procedure is repeated unless r_0 is assigned for all the rows. Hence, both the windows W_1 and W_{2n-2} become conflict-free.

The same procedure is to be followed to assign next AR-bit r_1, making the windows W_2 and W_{2n-3} conflict free.

In this way, considering the $(n-1)$ AR-bits, we necessarily can make all the $(2n-2)$ windows conflict-free which completes the conflict free routing of the given permutation P.

It is to be noted that for any arbitrary permutation, the procedure always can find a conflict-free routing. Hence, the network is rearrangeable. It proves the theorem. □

Remark 6. Example 11 shows that Benes network satisfies Theorem 1. Obviously, all the combined $(2n-1)$-stage networks which satisfy Theorem 1, are rearrangeable.

Example 12. Fig. 7 shows an $\Omega^{-1}(3)$ network, with top-to-bottom labeling. The shuffle at the output is ignored, since it just causes some relabeling at the output. The corresponding *i-mappings* are given by

f_0: $x_2x_1r_0$, f_1: $x_0x_2r_1$, f_2: $x_0x_2r_2$.

Figs. 2 and 7 show that for $\Omega \oplus \Omega^{-1}$, the concatenation mapping is an identity permutation. Hence the *i-mappings* for $\Omega \oplus \Omega^{-1}$ are given as:

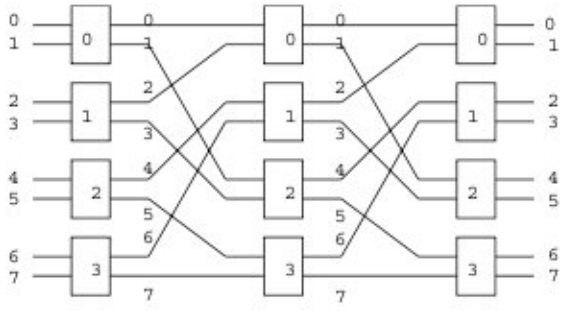

Fig. 7. An 8×8 inverse omega network $\Omega^{-1}(3)$.

F_0: $x_1x_0r_0$, F_1: $x_1x_0r_1$, F_2: $x_1x_0r_2$, F_3: $x_0x_2r_3$, F_4: $x_0x_2r_4$.

Any path $x \rightarrow y$ in the network, where x and y in binary are $(x_2x_1x_0)$ and $(y_2y_1y_0)$, respectively, is represented by the sequence of links:

$$x_2x_1x_0 \xrightarrow{F_0} x_1x_0r_0 \xrightarrow{F_1} x_0r_0r_1 \xrightarrow{F_2} r_0r_1r_2 \xrightarrow{F_3} r_2r_0r_3 \xrightarrow{F_4} r_3r_2r_4.$$

Hence, the corresponding characteristic strings for the windows are

S_0: $x_2x_1x_0$, S_1: $x_1x_0r_0$, S_2: $x_0r_0r_1$, S_3: $r_0r_1r_2$, S_4: $r_2r_0r_3$, and S_5: $r_3r_2r_4$. These strings satisfy Theorem 1. Hence the network is rearrangeable.

Example 13. Fig. 2 shows that for $\Omega \oplus \Omega$ network, the concatenation mapping is an identity permutation. Hence the i-mappings for $\Omega \oplus \Omega$ are given as:

F_0: $x_1x_0r_0$, F_1: $x_1x_0r_1$, F_2: $x_1x_0r_2$, F_3: $x_1x_0r_3$, and F_4: $x_1x_0r_4$.

Any path $x \rightarrow y$ in the network, where x and y in binary are $(x_2x_1x_0)$ and $(y_2y_1y_0)$, respectively, is represented by the sequence of links:

$$x_2x_1x_0 \xrightarrow{F_0} x_1x_0r_0 \xrightarrow{F_1} x_0r_0r_1 \xrightarrow{F_2} r_0r_1r_2 \xrightarrow{F_3} r_1r_2r_3 \xrightarrow{F_4} r_2r_3r_4$$

Hence, the corresponding characteristic strings for the windows are

S_0: $x_2x_1x_0$, S_1: $x_1x_0r_0$, S_2: $x_0r_0r_1$, S_3: $r_0r_1r_2$, S_4: $r_1r_2r_3$, and S_5: $r_2r_3r_4$. These strings do not satisfy Theorem 1. Hence the rearrangeability of the network can not be guaranteed.

The networks $\beta \oplus \beta$, $\Omega \oplus \beta$, $\Omega \oplus \beta^{-1}$, $\beta \oplus \Omega^{-1}$, $\beta^{-1} \oplus \Omega^{-1}$ are some more examples of combined networks which satisfy Theorem 1, and therefore rearrangeable.

4.2. Algorithms for checking rearrangeability

Given any Ω-equivalent network $M(n)$, here follows the algorithm to label the switches and the links, and to find out the corresponding i-mappings. As has been mentioned earlier, the equivalence with Ω network can be checked in $O(Nn)$ time using the algorithm presented in [6]. If $M(n)$ is equivalent to Ω network, it is obvious that it is also equivalent to baseline network $\beta(n)$. In the following algorithm, the switches are labeled according to the one-to-one correspondence between the nodes of $M(n)$ and $\beta(n)$.

Let any switch of $M(n)$ be identified as $S(i,j)$, where i is the stage, and j is the physical position of it in stage i, $0 \leqslant i \leqslant (n-1)$, and $0 \leqslant j \leqslant (N/2-1)$. The input links of $S(i,j)$ are represented as $l_I(i,j)$, and $l'_I(i,j)$, and the corresponding output links as $l_O(i,j)$ and $l'_O(i,j)$, respectively, such that if the switch is set straight the link $l_I(i,j)$ is connected to $l_O(i,j)$, and link $l'_I(i,j)$ is connected to $l'_O(i,j)$.

The following algorithm marks the nodes $S(i,j)$ of each stage uniquely by the labels $(0,1,\ldots,N/2-1)$ and the two input (output) links of each node are marked as '0' and '1', respectively. The node connected by the output link marked as '0' ('1') of $S(i,j)$ in the next stage is called the *u-child* (*d-child*). Similarly, the node connected by the input link marked as '0' ('1') of $S(i,j)$ in the previous stage is called the *u-parent* (*d-parent*).

Algorithm (Label-Switch)

Input. The n-level graph of $M(n)$ with nodes $S(i,j)$, $0 \leqslant i \leqslant (n-1)$, and $0 \leqslant j \leqslant (N/2-1)$.
Output. The set of nodes $\{S(i,j)\}$, each with a label $x = x_{n-2}x_{n-3}\ldots x_0$, $0 \leqslant x \leqslant (N/2-1)$, and two input (output) links marked as '0' and '1', respectively.
Step 1. Mark $S(0,0)$ as $0, l_I(0,0) = l_O(0,0) = 0$, and $l'_I(0,0) = l'_O(0,0) = 1$.
Step 2. For $i = 0$ to $(n-2)$ do
{for $j = 0$ to $(N/2-1)$ do
{if $S(i,j)$ is marked as $x = x_{n-2}x_{n-3}\ldots x_0$
then mark u-child of $S(i,j)$ as $x' = x_{n-2}\ldots x_{n-i-1}0x_{n-i-2}\ldots x_1$, and
mark d-child of $S(i,j)$ as $x'' = x_{n-2}\ldots x_{n-i-1}1x_{n-i-2}\ldots x_1$.

If x is even, the connecting input link and corresponding output link of both u-child and d-child are marked as '0', and the other pair of input-output links are marked as '1'.
else the marks on links are reversed.}}
Step 3. For $i = (n-1)$ to 1 step (-1) do
{for $j = 0$ to $(N/2-1)$ do
{for $S(i,j)$ marked as $x = x_{n-2}x_{n-3}..x_0$, if any parent of it is unmarked
then for u-parent mark it as $x' = x_{n-2}..x_{n-i}\,x_{n-i-2}\ldots x_0 0$, and
for d-parent mark it as $x'' = x_{n-2}x_{n-i}x_{n-i-2}\ldots x_0 1$.
If x_{n-i-1} bit in x is '0',
the connecting output link and corresponding input link of both u-parent and d-parent are marked as '0', and the other pair of input-output links are marked as '1'.
else the marks of links are reversed.}}
Step 4. Terminate

The above algorithm assigns a unique label x, $0 \leqslant x \leqslant (N/2-1)$ to each switch $S(i,j)$ of any stage i, $0 \leqslant i \leqslant (n-1)$, and marks the two input (output) links of each switch as $m = 0$ and $m = 1$, respectively.

Example 14. Fig. 8 shows the switch labels and m values on links of the MIN $M(3)$ as derived by *Algorithm Label Switch*.

Note that Fig. 3 shows the same network with labels according to its topological isomorphism with $\Omega(3)$, whereas in Fig. 8 the *Algorithm Label Switch* labels the switches according to $\beta(3)$.

Here follows another algorithm that runs on the output of *Algorithm Label Switch*, and assigns unique label to each input of $M(n)$, and to each output link of any stage i, $0 \leqslant i \leqslant (n-1)$. Hence it computes the *i-mappings*of the network.

Algorithm (i-Mappings)

Input. The network $M(n)$ with nodes $S(i,j)$, $0 \leqslant i \leqslant (n-1)$, and $0 \leqslant j \leqslant (N/2-1)$, each with a label x, $0 \leqslant x \leqslant (N/2-1)$, and two input (output) links marked as $m = 0$ and $m = 1$, respectively.
Output. The set of *i-mappings* f_i, each an array of size n, representing a permutation of $(0,1,\ldots,(n-1))$, $0 \leqslant i \leqslant (n-1)$.

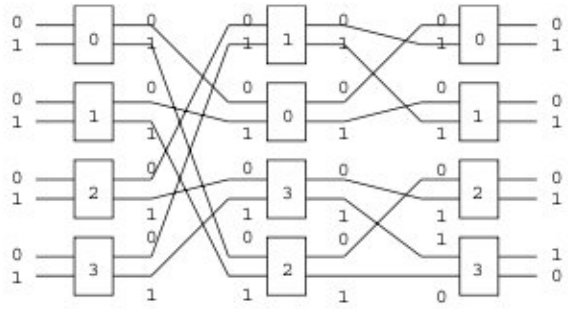

Fig. 8. $M(3)$ with switch labels and link labels as derived from algorithm *label switch*.

Step 1. For each input switch $S(0,j)$ label upper (lower) input link as $2j$ $(2j+1)$.
For each output switch $S(n-1,j)$ label upper (lower) output link as $2j$ $(2j+1)$.
Step 2. For $i = 0$ to $(n-2)$ do
{for $j = 0$ to $(N/2-1)$ do
{for each switch $S(i,j)$ label output links as $(2 * label\ of\ S(i,j) + m)$}}
Step 3. $f_0(0) := r_0$
For $j = 0$ to $(N-1)$ do
{for any input link of $S(0,j)$ with label 2^k, $k = 0, 1, \ldots, (n-1)$, if the label of corresponding output link of $S(0,j)$ is 2^p, $p \neq 0$, then $f_0(p) := k$, else 'no i-mappings' and terminate.}
Step 4. For $i = 0$ to $(n-2)$ do
{$f_i(0) := r_i$
for $j = 0$ to $(N-1)$ do
{if the label of the link l_O of $S(i,j)$ is 2^k, $k = 0, 1, \ldots, (n-1)$, and the label of connecting link at stage $(i+1)$ is 2^p, $p \neq 0$, then $f_{i+1}(p) := k$, else 'no i-mappings'.}}
Step 5. Terminate

The above algorithm finds f_i, $0 \leqslant i \leqslant (n-1)$ as an array of size $(n-1)$, where $f_i(p) = k$, $0 \leqslant k \leqslant (n-1)$, $1 \leqslant p \leqslant (n-1)$ denotes that in the mapping f_i the kth bit of input string is mapped to the pth bit of output string, and $f_i(0) = r_i$.

Example 15. Fig. 9 shows the link labels of the MIN $M(3)$ as derived by *Algorithm i-Mappings*.

The algorithm also outputs the following set of *i-mappings* for $M(3)$:

f_0: $x_2x_1r_0$, f_1: $x_0x_2r_1$, and f_2: $x_2x_0r_2$.

Note that these are same as $\beta(3)$.

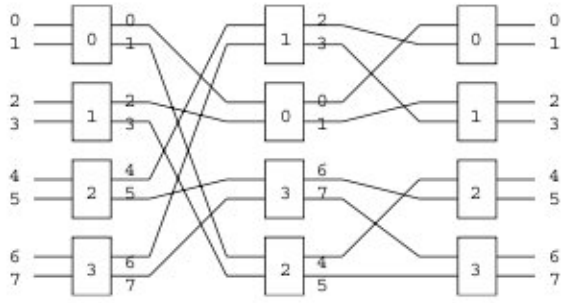

Fig. 9. $M(3)$ with switch labels and link labels as derived from algorithm *i-mappings*.

Therefore, given any two Ω-equivalent networks Δ and Δ', the *i*-mappings of the networks can be found along with a basic labeling of the links by the algorithms described above.

Now let us consider the combination of two such networks Δ and Δ' to form the network $\Delta \oplus \Delta'$. The following procedure will check whether this network satisfies Theorem 1 or not.

Algorithm (Check-rearrangeability)

Step 1. For each network Δ and Δ', apply *Algorithm Label Switch*, and then *Algorithm i-Mappings* to obtain a basic labeling and corresponding *i*-mappings $\{f_i\}$ and $\{f_i'\}$, respectively, $0 \leqslant i \leqslant (n-1)$.

Step 2. For each output switch $S(n-1,j)$ of Δ, if the input switch of Δ' is switch $S'(0,k)$, $C(2j) := 2k$ and $C(2j+1) := 2k+1$ for $0 \leqslant j \leqslant N/2-1$.

Step 3. If $C(2^p) = 2^m$, for all p, $0 \leqslant p, m \leqslant (n-1)$, and for any $i = x_{n-1}x_{n-2}\ldots x_1x_0$, $0 \leqslant i \leqslant N-1$, if $C(i) = \Sigma x_j \cdot C(x_j)$, for all j, $0 \leqslant j \leqslant N-1$, go to next step, else report success := 0 and terminate.

Step 4. $F_i := f_i$, for $0 \leqslant i \leqslant (n-2)$, $F_i := f_i'$, for $n \leqslant i \leqslant (2n-2)$, and $F_i := C \circ f_i$, for $i = (n-1)$.

Step 5. $S_0 := x_{n-1}x_{n-2}\ldots x_1x_0$, $S_k := F_{k-1}[S_{k-1}]$ for $1 \leqslant k \leqslant 2n-1$.

Step 6. For $i = 0$ to $(n-2)$ do
if $r_i \in S_j$, for all j, $(i+1) \leqslant j \leqslant (2n-2-i)$ and $r_i \notin S_j$ for all j, $(2n-1-i) \leqslant j \leqslant (2n-1)$, report success := 1 else success := 0 and terminate

If the procedure reports success := 1, we know that the combined network satisfies Theorem 1, and is therefore rearrangeable.

Example 16. Fig. 10 shows the network $M(3) \oplus \beta(3)$. The switch labels and the *i*-mappings for $M(3)$ are same as given in Example 15, whereas for $\beta(3)$ those are given in Example 2.

In this case, the concatenation mapping C is an identity permutation.

The *characteristic strings* for $M(3) \oplus \beta(3)$ as obtained by the above algorithm are

S_0: $x_2x_1x_0$, S_1: $x_2x_1r_0$, S_2: $r_0x_2r_1$, S_3: $r_0r_1r_2$, S_4: $r_2r_0r_3$, and S_5: $r_2r_3r_4$.

Here the AR-bits are r_0 and r_1, respectively. It satisfies Theorem 1. Hence the network is rearrangeable.

Each of the three algorithms mentioned above is of complexity $O(Nn)$. Therefore, given any two Ω-equivalent networks Δ and Δ', it can be checked whether $\Delta \oplus \Delta'$ satisfies Theorem 1 and hence is rearrangeable, in $O(Nn)$ time.

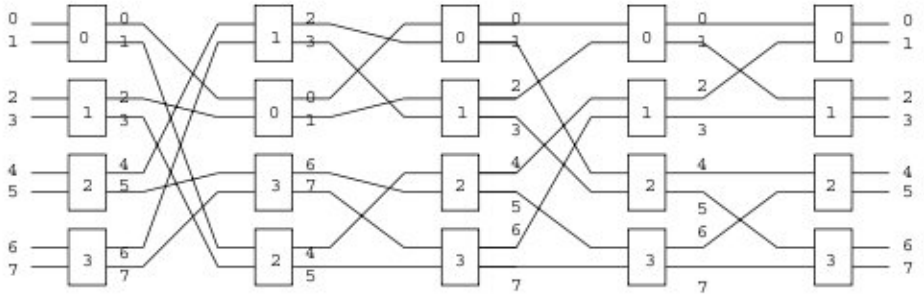

Fig. 10. Combined network $M(3) \oplus \beta(3)$.

4.3. Routing in combined networks

By the technique presented here, if it is found that a given combined network satisfies Theorem 1, and hence is rearrangeable, we can easily assign the AR-bits deterministically to route any permutation without any conflict, following the same procedure, as has been described in the proof of Theorem 1. Here follows the algorithm.

Algorithm (Routing)

Input. The characteristic strings S_j (an n-bit array), for each window W_j, $1 \leqslant j \leqslant (2n-2)$, and the windows W_0 and W_{2n-1}, i.e., the input and output windows, respectively, derived from the given permutation P.

Output. $(n-1)$ AR-bits $r_0, r_1, \ldots, r_{n-2}$, each represented as an N-bit array, where $r_k(j)$, $0 \leqslant k \leqslant (n-2)$, $0 \leqslant j \leqslant (N-1)$ represents the routing bit for input j at stage k.

For $k = 0$ to $(n-2)$ do

{For any row j, $0 \leqslant j \leqslant (N-1)$, $r_k(j) := 0$; find the conjugate row j', in W_k.

$r_k(j') := 1$; find the conjugate row j'' in W_{2n-1-k}.

$r_k(j'') := 0$.

Repeat it until r_k bit of a conjugate row is found to be already assigned.

If all r_k's are not filled up, start from any arbitrary unfilled row and repeat the procedure until all rows are filled up.

The windows W_{k+1} and W_{2n-2-k} are formed defined by S_{k+1} and S_{2n-2-k}, respectively.}

This algorithm will determine the AR-bits for each path for routing through first $(n-1)$ stages, for the rest each path is self-routing, i.e., the destination tag itself will determine the route. As has been explained in Theorem 1, the above procedure will always be successful to find conflict-free paths for any arbitrary permutation through a combined network $\Delta \oplus \Delta'$, if the network satisfies Theorem 1. It is evident that the time complexity of the algorithm is $O(Nn)$ only.

Example 17. Consider the combined network $M(3) \oplus \beta(3)$ shown in Fig. 10. Given any permutation

$$P : \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 3 & 1 & 4 & 7 & 2 & 5 & 0 & 6 \end{pmatrix}$$

the input and output windows W_0 and W_5 are as follows:

$$\begin{pmatrix} \leftarrow & W_0 & \rightarrow & \leftarrow & W_5 & \rightarrow \\ x_2 & x_1 & x_0 & y_2(r_2) & y_1(r_3) & y_0(r_4) \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 \end{pmatrix}$$

Now, the *Routing Algorithm* determines the AR-bits in the respective windows as shown below

$$\begin{pmatrix} \leftarrow & W_1 & \rightarrow & & \\ & & \leftarrow & W_4 & \rightarrow \\ x_2 & x_1 & r_0 & y_2(r_2) & y_1(r_3) \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

$$\begin{pmatrix} \leftarrow & W_2 & \rightarrow & \\ & \leftarrow & W_3 & \rightarrow \\ x_2 & r_0 & r_1 & y_2(r_2) \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{pmatrix}$$

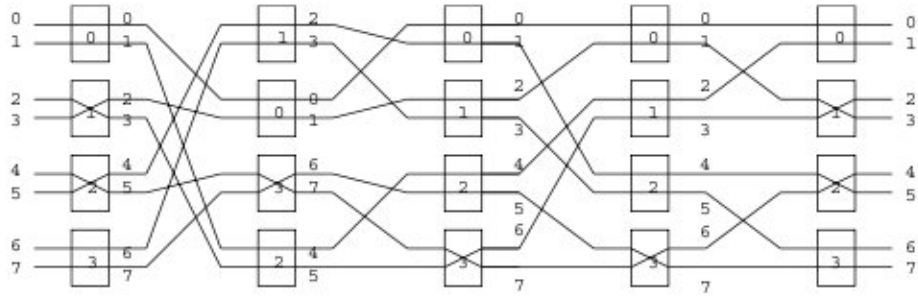

Fig. 11. Paths through $M(3) \oplus \beta(3)$ for realizing P.

For the remaining stages the paths follow the destination tag routing. The complete paths are shown in Fig. 11.

5. Concatenation for rearrangeability

With the ideas developed so far, given two Ω-equivalent networks Δ and Δ', if they are concatenated by superimposing the switches according to their physical positions only, as has been considered so far, the rearrangeability of the combined network $\Delta \oplus \Delta'$ can not be guaranteed. If $\Delta \oplus \Delta'$ is found to be equivalent to Benes network, it is obviously rearrangeable. But so far it needs $O(N^4 n)$ time to check it. More interestingly, many combined networks would not be equivalent to Benes network and hence their rearrangeability is not guaranteed.

By our technique, if the concatenation mapping be a BP-permutation, we can determine whether the network satisfies Theorem 1, and hence rearrangeable, in $O(Nn)$ time. However, again, the rearrangeability of any such $\Delta \oplus \Delta'$ is never assured, even when the concatenation mapping is a BP permutation.

But most interestingly, if we modify the combining procedure a little bit, we can guarantee the rearrangeability of the combined network always.

5.1. Concatenation technique

Here, the $\beta \oplus \beta$ network is taken as the standard example of combined network which is rearrangeable.

Given two Ω-equivalent networks Δ and Δ' the steps followed for concatenation are stated below:

- Label the switches of each network according to the one-to-one correspondence with baseline network $\beta(n)$ by *Algorithm label-switch*. These labels are logical ID's of the switches at each stage. Find the i-mappings for each.
- Concatenate the two networks by superimposing the switches with same logical ID's. The concatenation mapping would be the identity mapping.
- Find the *i-mappings* for the combined network by *Algorithm i-Mapping*. It would be similar to that of $\beta(n) \oplus \beta(n)$, except some relabeling at the final input and/ or output links. Hence it will satisfy Theorem 1, and will be rearrangeable.

The new network is represented as $\Delta \odot \Delta'$. It is to be noted that the whole procedure would be completed in $O(Nn)$ time.

Example 18. We know that $\Omega \oplus \Omega$ does not satisfy Theorem 1. But the combined network $\Omega \odot \Omega$ satisfies Theorem 1, and hence is rearrangeable.

Fig. 12 shows the one-to-one correspondence of the switches of $\Omega(3)$ with $\beta(3)$, as given by algorithm *Label Switch*. Fig. 13 shows the logical combination $\Omega(3) \odot \Omega(3)$. In the combined network, any path $x \to y$, where $x_2 x_1 x_0$, and $y_2 y_1 y_0$ are the binary representations of x and y, respectively, can be represented at different windows w_j, for $0 \leqslant j \leqslant 5$ as:

$$x_2 x_1 x_0 \to\ x_0 x_1 r_0 \to r_0 x_0 r_1 \to r_0 r_1 y_2 \to y_2 r_0 y_1 \to y_2 y_1 y_0.$$

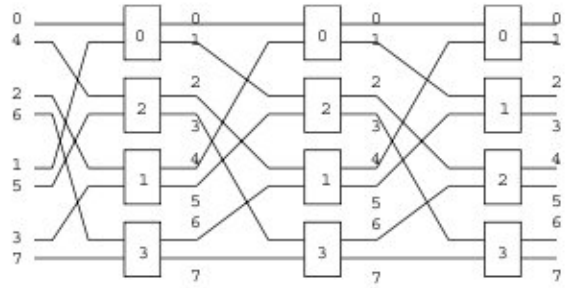

Fig. 12. $\Omega(3)$ with switch labels and link labels according to $\beta(3)$.

It satisfies Theorem 1, and hence the network is rearrangeable.

Theorem 2. *The combination of any two Ω-equivalent network, say $\Delta \odot \Delta'$ always can be made rearrangeable, if we label the switches of Δ and Δ' according to the isomorphism with two known topologies, say M and M′, respectively, and it is known that $M \oplus M'$ is rearrangeable.*

Proof. If the switches and links of Δ network are labeled according to its isomorphism with network M, the *i-mappings* of Δ network will be same as those of M network. Similarly, the *i-mappings* of Δ' will be same as those of M' network. The superposition is done according to the logical-ID's of the switches. Hence, the concatenation mapping is an identity permutation. Therefore, the *i-mappings*, and hence the characteristic strings of the windows for network $\Delta \odot \Delta'$ will be the same as those of $M \oplus M'$. It proves the theorem. □

5.2. Difference between Benes and $\Omega \oplus \Omega$ networks

With the ideas developed so far, finally the difference between the topologies of Benes and $\Omega \oplus \Omega$ networks can be pointed out which actually causes the behavioral difference of the two.

The switch labels and link labels of the $\Omega(3)$ network according to *Algorithm label-switch* are shown in Fig. 12. When another $\Omega(3)$ is concatenated with it superimposing the switches in the

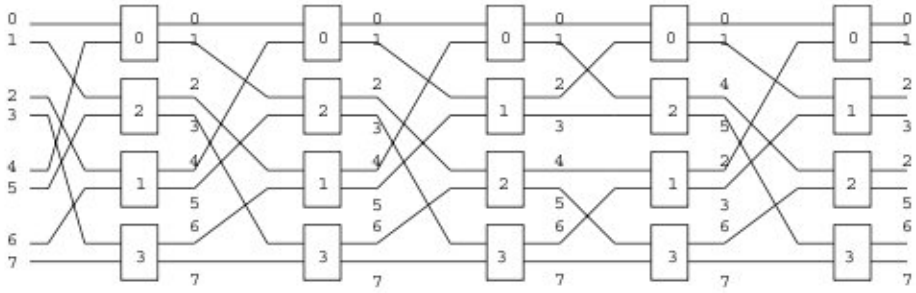

Fig. 13. Network $M(3) \odot \beta(3)$.

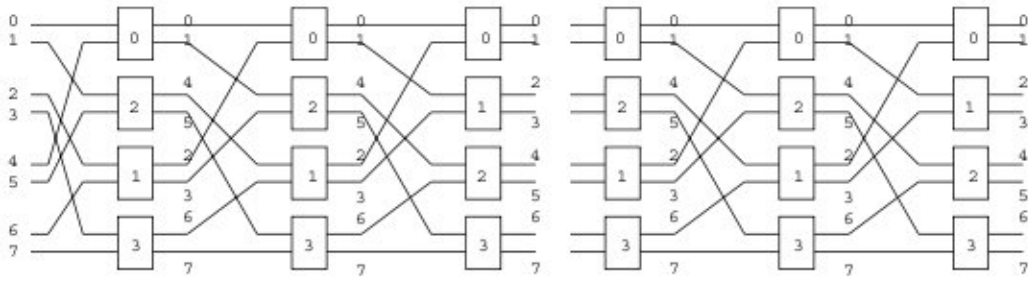

Fig. 14. Two $\Omega(3)$ networks to be combined to form $\Omega(3) \oplus \Omega(3)$.

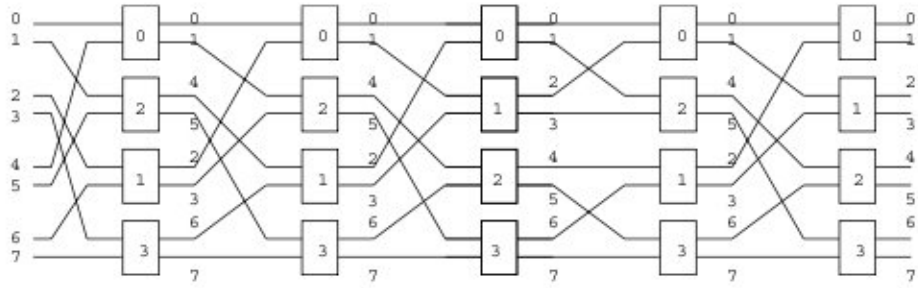

Fig. 15. Combined network $\Omega(3) \odot \Omega(3)$.

first stage of $\Omega(3)$ with the corresponding one in the last stage of $\Omega(3)$, at the same physical position, it is evident that switches merged are not with the same logical-ID. Fig. 14 shows the concatenation of two $\Omega(3)$ networks to form $\Omega(3) \oplus \Omega(3)$ network. Here, switch with logical-ID 2 is merged with switch labeled as 1, and vice versa. But from Fig. 6, it can be noted that in Benes network which is the concatenation of two Ω-equivalent networks, the switches with same logical-ID are superimposed. In fact, this difference in the topologies of Benes network and $\Omega \oplus \Omega$ actually causes the behavioral difference of the two networks. Example 13 shows that the $\Omega \oplus \Omega$ does not satisfy Theorem 1. Hence, the rearrangeability of $\Omega \oplus \Omega$ network still remains an open problem.

More interestingly, Fig. 15 shows the concatenation of two $\Omega(3)$ networks to form $\Omega(3) \odot \Omega(3)$ network. Here, the switch pairs merged have the same logical-ID, according to the topological isomorphism with $\beta(3)$. Hence by Theorem 2, this network would be a rearrangeable network.

6. Conclusion

So far, a lot of research has been reported on $(2n-1)$-stage combined networks $M \oplus M'$, formed by concatenating two n-stage Ω-equivalent networks M and M', respectively. But there is no algorithm, in general, to check whether a given combined network is rearrangeable or not. Though both Benes network and $\Omega \oplus \Omega$ belong to this class, the former one is a rearrangeable network whereas the rearrangeability of the latter one is still an open problem. In this paper, a sufficient condition for the rearrangeability of a given combined network has been established. An $O(Nn)$ algorithm has been presented for routing in such a rearrangeable combined network. Moreover, a novel scheme has been presented for concatenating any two Ω-equivalent networks that always results a rearrangeable network. Finally, it points out the exact difference between the topologies of Benes network and $\Omega \oplus \Omega$ network.

References

[1] Y. Pan, C. Qiao, Y. Yang, Optical multistage interconnection networks: new challenges and approaches, IEEE Commun. Mag. 37 (2) (1999) 50–56.

[2] V.E. Benes, On rearrangeable three-stage connecting networks, The Bell Sys. Technical J. XLI (5) (1962) 1481–1492.

[3] D. Nassimi, S. Sahni, A self-routing Benes network and parallel permutation algorithms, IEEE Trans. Comput. C 30 (5) (1982) 332–340.

[4] C.S. Raghavendra, R.V. Boppana, On self-routing in Benes and shuffle-exchange networks, IEEE Trans. Comput. 40 (1991) 1057–1064.

[5] N. Das, K. Mukhopadhyaya, J. Dattagupta, O(n) routing in rearrangeable networks, J. Syst. Architect. 46 (4) (2000) 529–542.

[6] Q. Hu, X. Shen, J. Yang, Topologies of combined $(2\log N-1)$-stage interconnection networks, IEEE Trans. Comput. 46 (1) (1997) 118–124.

[7] Y. Yang, J. Wang, Y. Pan, Permutation capability of optical multistage interconnection networks, J. Parallel Distribut. Comput. 60 (2000) 72–91.

[8] D.P. Agrawal, Graph theoretical analysis and design of multistage interconnection networks, IEEE Trans. Comput. 32 (7) (1983) 637–648.

[9] K.Y. Lee, On the rearrangeability of $(2\log N-1)$-stage permutation networks, IEEE Trans. Comput. C 34 (5) (1985) 412–425.

[10] Y. Yeh, T. Feng, On a class of rearrangeable networks, IEEE Trans. Comput. 41 (11) (1992) 1361–1379.

[11] H. Cam, J.A.B. Fortes, Work-efficient routing algorithms for rearrangeable symmetric networks, IEEE Trans. Parallel Distribut. Syst. 10 (7) (1999) 733–741.
[12] T. Feng, S.-W. Seo, A new routing algorithm for a class of rearrangeable networks, IEEE Trans. Comput. 43 (11) (1994) 1270–1280.
[13] C.L. Wu, T. Feng, On a class of multistage interconnection networks, IEEE Trans. Comput. C 29 (8) (1980) 694–702.
[14] T. Feng, A survey of interconnection networks, IEEE Comput. Mag. 14 (8) (1981) 12–27.
[15] X. Shen, An optimal O(Nlg$N$) algorithm for permutation admissibility to extra-stage cube-type networks, IEEE Trans. Comput. 44 (9) (1995) 1044–1049.
[16] D.C. Opferman, N.T. Tsao-Wu, On a class of rearrangeable switching networks—part I: control algorithm, Bell Syst. Techn. J. 50 (5) (1971) 1579–1600.
[17] C.S. Raghavendra, A. Varma, Fault-tolerant multiprocessors with redundant-path interconnection networks, IEEE Trans. Comput. C 35 (4) (1986) 307–316.
[18] A. Varma, C.S. Raghavendra, Rearrangeability of multistage shuffle/exchange networks, IEEE Trans. Commun. 36 (10) (1988) 1138–1143.

Nabanita Das received the B.Sc. (Hons.) degree in Physics in 1976, B.Tech. in Radio Physics and Electronics in 1979, from the University of Calcutta, the M.E. degree in Electronics and Telecommunication Engineering in 1981, and Ph.D in Computer Science in 1992, from Jadavpur University, Kolkata. Since 1986, she has been on the faculty of the Advanced Computing and Microelectronics unit, Indian Statistical Institute, Calcutta. She visited the department of Mathematik and Informatik, University of Paderborn, Germany, under INSA scientists' exchange programme. She has co-authored many papers published in International journals of repute. She has acted as the co-guest editor of the special issue on 'Resource Management in mobile, ad hoc and sensor networks' of Microprocessors and Microsystems, by Elsevier. She has served as a member of the program committee of different international conferences in the area of Distributed Computing, High performance computing and Communication. Presently, she is acting as program chair of International workshop on distributed computing, IWDC 2004, and also as co-editor of the proceedings to be published as LNCS by Springer. Her research interests include parallel processing, interconnection networks, wireless communication and mobile computing. She is a senior member of IEEE.