Rearrangeability of a Class of $(2 \log N - 1)$ Stage Networks Using the LCP Based Decomposition

Annalisa Massini Maria Teresa Raffa

Dipartimento di Informatica, Sapienza Università di Roma,

via Salaria 113 - 00198 Rome, Italy

Abstract

In this paper we describe a routing algorithm that routes any permutation on a $(2 \log N-1)$ stage interconnection network in $O(N \log N)$ time. The proposed algorithm works on any multistage interconnection network, MIN, belonging to the equivalence class represented by the concatenation of a Reverse Butterfly and a Butterfly whose first and second stages are swapped. Both the routing algorithm and the definition of equivalence classes are based on the decomposition in factors of MINs obtained by using the Layered Cross Product. The interest of this result is its approach, that is based on the use of only one factor of the studied MIN. Moreover, the algorithm provides a proof that all $(2 \log N - 1)$ stage MINs obtained concatenating two $\log N$ stage Butterfly equivalent MINs, with $N = 8$ inputs are rearrangeable.

I. Introduction

Interconnection networks are widely studied for realizing communication among processors and for distributing information in telecommunication systems, using both electronic and optical technologies. Many interconnection network topologies have been considered in the past decades and, among these, Multistage Interconnection Networks, MINs, offer a good trade-off between routing time complexity and topological complexity. An important requirement on interconnection networks is the realizability of any permutations of data between inputs and outputs. An N input MIN is called a *rearrangeable network* if it realizes every one of the $N!$ permutations in a single pass. MINs consisting of $\log N$ stages such as Omega, Flip, Baseline and Reverse Baseline, Butterfly and Reverse Butterfly are all equivalent networks [4], [5] and have attractive features, but they are not rearrangeable. For this reason, MINs obtained by concatenating two $\log N$ stage MINs with the center stage overlapped, have been intensively studied. Indeed, $2 \log N - 1$ is the theoretically minimum number of stages required for obtaining rearrangeable multistage interconnection networks [10]. The popular $(2 \log N - 1)$ stage Beneš network [2], [3] is rearrangeable and the Looping algorithm provides a method and a proof for its rearrangeability. Unfortunately it can be used only on $(2 \log N - 1)$ stage symmetric MINs with recursive structure such as Baseline-Reverse Baseline and Butterfly-Reverse Butterfly networks. The Looping algorithm does not work on the Omega-Omega^{-1} or Double Baseline even if they are in the class of Beneš equivalent networks as shown in [5]. For the Omega-Omega^{-1}, Lee [11], [12] proposed a routing algorithm based on the decomposition of the first half of the network and the self-routing on the second half. The main problem of these two schemes is that they exploit the network topology configuration and then work only on the networks they are designed for. Both this algorithm and the Looping algorithm have $O(N \log N)$ time complexity. Recently, Das in [7] has formulated a sufficient condition for checking the rearrangeability of $(2 \log N - 1)$ stage MINs and has presented a routing algorithm requiring an $O(N \log N)$ time complexity. Note that lower values for the time complexity have been obtained only for special class of permutations.

In [6] an algorithm to route any permutation on any Beneš equivalent MIN is described. In that work the Layered Cross Product, LCP, of Even and Litman [8], is exploited to obtain the decomposition of $(2 \log N - 1)$ stage MINs in factors. It is possible to decompose MINs either in bamboos [8], [5], [9] exploiting the theory of prime factors of Paz [13], or in binary trees [8], [5]. In both cases, it is possible to exhibit a partition of $(2 \log N - 1)$ stage MINs in equivalence classes [5]. The decomposition in bamboos has been recently used in [1], by Bao et al., for Bit Permutation Networks, BPN, that are MINs where the number of stages is not fixed, and a necessary condition for a BPN to be rearrangeable is given.

In this work we describe an algorithm that routes any permutation on a MIN belonging to the equivalence class represented by the concatenation of a Reverse Butterfly and a Butterfly whose first and second stages are swapped, exploiting the decomposition factors of the network.

The interest of this result is in the approach used, that is based on the use of only a factor of the studied MIN, in a way similar to that presented in [6]. Moreover, the proposed algorithm provides a proof that all $(2\log N - 1)$ stage MINs obtained by concatenating two $\log N$ stage MINs, with $N = 8$ inputs are rearrangeable. Finally, algorithms using the LCP based decomposition allow a deep understanding of features of MINs and, hopefully, represents a step towards the proof of the rearrangeability of Omega-Omega, that still remains an open problem for $N = 2^n, n > 3$ [15].

II. Preliminaries

In this section we give some definitions and introduce some notations used in the rest of the paper.

Definition II.1. *A* permutation Π for a MIN *is a set of N different input-output pairs (i, j) with $i, j \in [1, N]$ having neither inputs nor outputs in common. Each pair represents the connection request between one input and one output.*

A MIN *satisfies* the permutation Π if there exists a set of N edge disjoint paths from the input to the output of each request in Π passing through exactly one node in each stage.

Definition II.2. *A MIN is* rearrangeable *if it can satisfy all the $N!$ permutations.*

The algorithm presented in this work is based on the partition of $(2\log N - 1)$ stage MINs in equivalence classes, described in [5]. This partition is obtained by means of the decomposition in factors of a MIN exploiting the theory of Layered Cross Product (LCP) [8]:

- an *l-layered graph*, $G = (V_1, V_2, \dots, V_l, E)$ consists of l layers of nodes, V_i is the (non-empty) set of nodes in layer i, where $1 \le i \le l$; E is a set of edges such that every edge connects nodes of two adjacent layers,
- the *Layered Cross Product*, $G = G' \otimes G''$, of two l-layered graphs $G' = (V_1', V_2', \dots, V_l', E')$ and $G'' = (V_1'', V_2'', \dots, V_l'', E'')$ is an l-layered graph $G = (V_1, V_2,$ $\dots, V_l, E)$ where V_i is the cartesian product of V_i' and V_i'', $1 \le i \le l$, and an edge $\langle (u', u''), (v', v'') \rangle$ belongs to E if and only if $\langle u', v' \rangle \in E'$ and $\langle u'', v'' \rangle \in E''$. G' and G'' are called the *first* and *second factor* of G, respectively.

A binary tree and a MIN are examples of layered graphs.

The operation of *decomposition in factors* is the inverse operation of the LCP. As described in [5], any $(2\log N - 1)$ stage MIN can be decomposed into two factors: the first one consists of two complete binary trees sharing their roots, call it $\substack{\triangledown \\ \vartriangle}$, and it is the same for all $(2\log N - 1)$ stage MINs; the other factor consists of two complete binary trees sharing their leaves, be it $\substack{\vartriangle \\ \triangledown}$, and it characterizes each equivalence class according to the way the leaves are connected. The number of the equivalence classes, that is the number of $\substack{\vartriangle \\ \triangledown}$, is $(\log N)!$.

In the case of $N = 8$, there are only two possible classes, that are the sub-$\substack{\vartriangle \\ \triangledown}$ of the first and second $\substack{\vartriangle \\ \triangledown}$ in Figure 1, on the left side. The first class contains the Beneš network and the Reverse Butterfly-Butterfly, the second class contains the Omega-Omega and the Butterfly-Butterfly (Reverse Butterfly-Reverse Butterfly).

All $(\log N)!$ possible equivalence classes in the case of $N = 16$ inputs (outputs) are shown in Figure 1 by means both of the $\substack{\vartriangle \\ \triangledown}$ factor and the representative MIN, visualized as a Reverse Butterfly concatenated

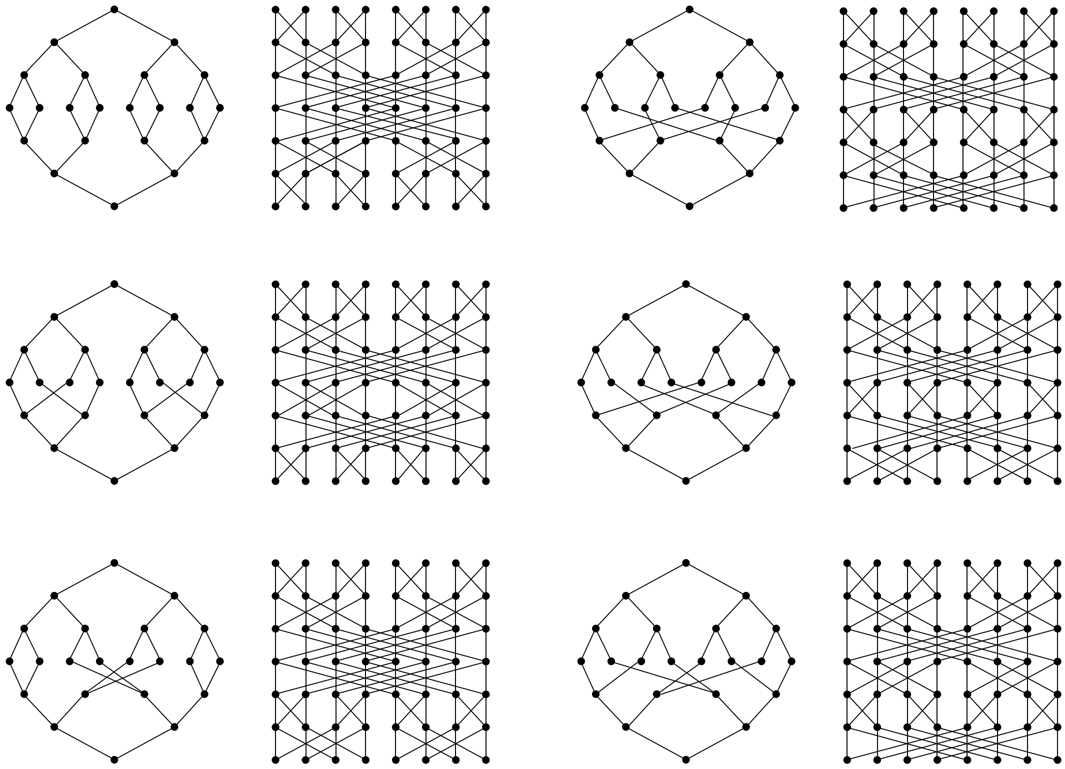

Fig. 1. All the possible $\overset{\triangle}{\triangledown}$ factors in the case of MINs with $N = 16$ inputs (outputs). MINs are represented using butterfly-like stages.

with a $\log N$ stage MIN obtained as any permutation of the Butterfly stages (factor $\overset{\triangledown}{\triangle}$ is not shown in Figure 1).

In [6] an algorithm for setting the switches of a Beneš equivalent MIN is described. It exploits only the $\overset{\triangle}{\triangledown}$ factor because the paths on the $\overset{\triangledown}{\triangle}$ factor are imposed by the tree structure. For this reason, the routing algorithm is designed by using only the $\overset{\triangle}{\triangledown}$ factor, but the relations among input or output elements, derived from $\overset{\triangledown}{\triangle}$, are taken into account to determine the paths. In particular, to avoid edge conflicts, elements (inputs or outputs) using the same path on $\overset{\triangledown}{\triangle}$, must be separated on factor $\overset{\triangle}{\triangledown}$, that is they must use different paths of $\overset{\triangle}{\triangledown}$. This can be obtained checking the building of the $\overset{\triangle}{\triangledown}$ paths level by level. To clarify this aspect we use Figure 2. Let us observe the position of the inputs of the two factors. The two elements of input pairs use the same edge on the first level of $\overset{\triangledown}{\triangle}$, then they must be associated to different edges of $\overset{\triangle}{\triangledown}$, namely if we associate input 1 to the right first level edge, then we must associate 2 to the left one, then if we use the left first level edge for 3, input 4 must use the right level, and so on. Of course, it is necessary to proceed in the choice of the left or right edge by taking into account which edges will be used on the other levels by each element.

For the second class of $(2\log N - 1)$ stage MINs of size $N = 8$, a general proof of rearrangeability is not available in literature. A proof of rearrangeability of a network in this class, that is the Five-Stage Shuffle/Exchange Network for $N = 8$, is due to Raghavendra and Varma [14] and is obtained by means of an algorithm providing the switch setting by building the pairs of input arriving to nodes of the middle stage. Once these pairs are formed, paths to middle stage nodes are provided and the self routing capability of the last $\log N$ stage is used to realize the requested permutation. As for the Beneš network algorithm, even this latter algorithm is applicable only to the network for which it is designed, since it is based on a scheme that exploits the network configuration, then it works only for the Five-Stage Shuffle/Exchange Network of size $N = 8$.

In next section, we describe a permutation routing algorithm for any MIN belonging to the second class of the case $N = 8$. This algorithm allows to prove that all MINs having a $\substack{\triangle\\\triangledown}$ factor with these sub-$\substack{\triangle\\\triangledown}$s are rearrangeable, that is all $(2\log N - 1)$ stage MINs obtained by concatenating a Reverse Butterfly and a Butterfly whose first and second stages are swapped, are rearrangeable.

III. The Permutation Routing Algorithm

In this section we propose a new permutation routing algorithm for any MIN belonging to the second class of the case $N = 8$, working on its $\substack{\triangle\\\triangledown}$ factor that in the following we denote ${\substack{\triangle\\\triangledown}}'_8$. Then we use the rearrangeability of this class of MINs to prove the rearrangeability of MINs that have ${\substack{\triangle\\\triangledown}}'_8$ as sub-$\substack{\triangle\\\triangledown}$ in their $\substack{\triangle\\\triangledown}$ decomposition factor.

The aim of the algorithm is to find suitable pairs of inputs and outputs that can reach the middle stage of ${\substack{\triangle\\\triangledown}}'_8$ without generating conflicts on edges. These pairs are inputs of the switches in the middle stage of the MIN. The requested permutation is obtained by using the self routing capability of $\log N$ stage MINs.

To this end, we define the following sequences that we use to check if input or output elements can be coupled avoiding conflicts generation.

Let Π be the permutation: $\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ \pi(1) & \pi(2) & \pi(3) & \pi(4) & \pi(5) & \pi(6) & \pi(7) & \pi(8) \end{pmatrix}$ and let UP and $DOWN$ be the sequences: $UP = [1,2,3,4,5,6,7,8]$ and $DOWN = [\pi(1),\pi(2),\pi(5),\pi(6),\pi(3),\pi(4),\pi(7),\pi(8)]$.

Observe that the sequence $DOWN$ is obtained from the considered permutation by swapping the pairs $(\pi(3),\pi(4))$ and $(\pi(5),\pi(6))$, due to the structure of $\substack{\triangle\\\triangledown}$.

We subdivide both the sequences, UP and $DOWN$, into two subsequences, that are: $UP_L = [1,2,3,4]$ and $UP_R = [5,6,7,8]$, $DOWN_L = [\pi(1),\pi(2),\pi(5),\pi(6)]$ and $DOWN_R = [\pi(3),\pi(4),\pi(7),\pi(8)]$.

In Figure 2 all the above defined sequences are depicted; the sequence UP consists of all inputs of $\substack{\triangledown\\\triangle}$ and $\substack{\triangle\\\triangledown}$; the sequence $DOWN$ consists of all outputs of $\substack{\triangledown\\\triangle}$ and $\substack{\triangle\\\triangledown}$; the sequences UP_L and UP_R consist of the elements on left and right edges entering in the middle node of $\substack{\triangledown\\\triangle}$. Analogously the sequences $DOWN_L$ and $DOWN_R$ consist of the elements on left and right edges outgoing from the middle node of $\substack{\triangledown\\\triangle}$. Note that it is not possible to visualize the sequences UP_L, UP_R, $DOWN_L$ and $DOWN_R$ on $\substack{\triangle\\\triangledown}$ because the position of each element on an edge depends on the given permutation. A further subdivision of the above sequences provides the following input pairs, p_{UP}, and output pairs, p_{DOWN}:

- $(1,2)$ and $(3,4)$ which form the sequence UP_L
- $(5,6)$ and $(7,8)$ which form the sequence UP_R
- $(\pi(1),\pi(2))$ and $(\pi(5),\pi(6))$ which form the sequence $DOWN_L$
- $(\pi(3),\pi(4))$ and $(\pi(7),\pi(8))$ which form the sequence $DOWN_R$

Observation 1. *Each successive subdivision of sequences UP and $DOWN$ provides the groups of elements on each edge level of $\substack{\triangledown\\\triangle}$ and allows us to check how to build groups of elements on $\substack{\triangle\\\triangledown}$ edge levels without conflicts. To avoid conflict generation, elements of the same group in $\substack{\triangledown\\\triangle}$ must be separated on $\substack{\triangle\\\triangledown}$, namely elements on the same edge of $\substack{\triangledown\\\triangle}$ must be associated to different edges in $\substack{\triangle\\\triangledown}$.*

We define the four paths in ${\substack{\triangle\\\triangledown}}'_8$, connecting the root of Δ to the root of ∇, by associating labels L, for left, and R, for right, to the edges of the two central stages of $\substack{\triangle\\\triangledown}$ as shown in Figure 2. Namely, the four paths are defined as follows:

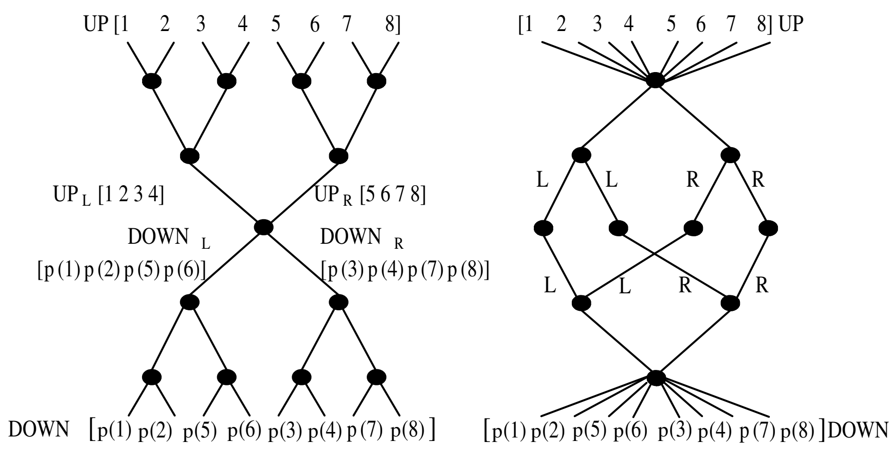

Fig. 2. Factors of a MIN of size $N = 8$ where sequences UP, $DOWN$, UP_L, UP_R, $DOWN_L$ and $DOWN_R$ are highlighted.

- P_{LL} uses the edges labeled L both in Δ and in ∇
- P_{LR} uses the edge labeled L in Δ and the edge labeled R in ∇
- P_{RL} uses the edge labeled R in Δ and the edge labeled L in ∇
- P_{RR} uses the edges labeled R both in Δ and in ∇

We introduce the notation $mateU(x)$ and $mateD(x)$ to indicate the relation $mate$ of coupling of an element x with the other element of the pair in sequences UP and DOWN respectively. It is easy to observe that for sequence UP the relation $mate$ is fixed, whereas for the sequence DOWN the relation depends on the considered permutation. As shown in the example of Figure 3, the relation $mate$ for element 3 provides: $mateU(3) = 4$ and $mateD(3) = 1$.

To form pairs that can be associated to paths of $\overset{\Delta}{\nabla}$ without generating edge conflicts, we require that for any pair (x, y) the following *Pair Properties* hold:

P1 $(x \in UP_L \wedge y \in UP_R) \vee (x \in UP_R \wedge y \in UP_L)$

P2 $(x \in DOWN_L \wedge y \in DOWN_R) \vee (x \in DOWN_R \wedge y \in DOWN_L)$

P3 pair (x, y) can be associated to fixed path P_{ij}, where $i, j \in \{L, R\}$ if and only if $((mateU(x) \notin P_{iz} \wedge mateU(y) \notin P_{iz}) \wedge (mateD(x) \notin P_{zj} \wedge mateD(y) \notin P_{zj}))$, where $z \in L, R$

Observation 2. *Pair Properties $P1$ and $P2$ avoid edge conflicts on level 2 and 3 of $\overset{\Delta}{\nabla}$. Pair Property $P3$ avoids edge conflicts on level 1 and 4 of $\overset{\Delta}{\nabla}$.*

To correctly form pairs satisfying *Pair Property* P3 for any permutation, we must avoid to couple elements that, even satisfying P1 and P2, can not be routed on $\overset{\Delta}{\nabla}'_8$ without conflicts, that is elements that can not be both associated to the same path. This can be obtained by choosing the first element of the first pair in a suitable way. This is realized by Step 1 of algorithm ROUTING ON $\overset{\Delta}{\nabla}'_8$, given in section III-B.

A. The Algorithm on an example for $\overset{\Delta}{\nabla}'_8$

Before giving the algorithm, we illustrate how it works by using the example in Figure 3. We determine the pairs of elements and the $\overset{\Delta}{\nabla}'_8$ path each pair is associated.

As an example, if permutation $\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 3 & 2 & 7 & 5 & 8 & 4 & 6 \end{pmatrix}$ is considered, see Figure 3, the sequence UP is $[1, 2, 3, 4, 5, 6, 7, 8]$, as usual, the sequence DOWN is $[1, 3, 5, 8, 2, 7, 4, 6]$, where pairs $(2, 7)$ and $(5, 8)$

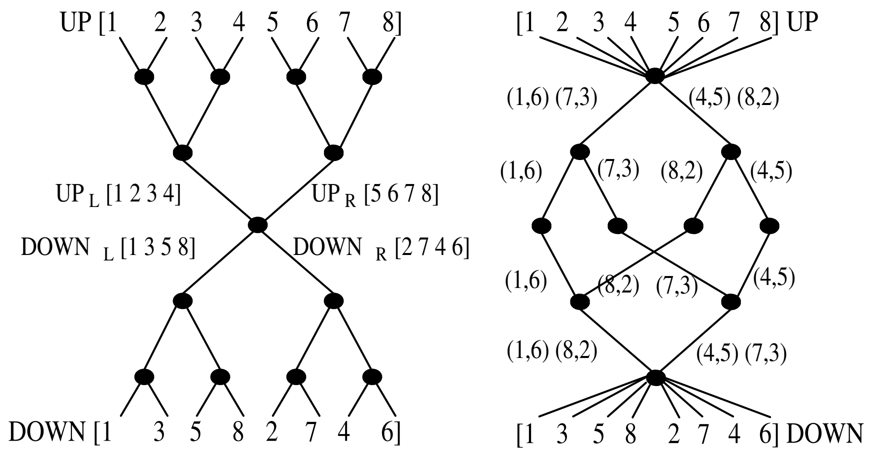

Fig. 3. The pairs founded by the algorithm for the middle node stage.

are swapped due to the structure of $\overset{\triangle}{\triangledown}$.

We obtain the following sequences, where pairs (and consequently the relation $mate$) are highlighted by means of parentheses:
$UP_L = [(1,2),(3,4)]$ and $UP_R = [(5,6),(7,8)]$
$DOWN_L = [(1,3),(5,8)]$ and $DOWN_R = [(2,7),(4,6)]$
Let us consider an element x_1 in the sequence UP as starting element. Usually, we start considering $x_1 = 1$, but this is not always possible (see *Procedure Check Permutation* in the following algorithm ROUTING ON $\overset{\triangle}{\triangledown}'_8$).

Then, in our example we start with $x_1 = 1$. To satisfy *Pair Property* P1, we must couple it with an element belonging to UP_R. We can not couple it with 5 because pair $(1,5)$ does not respect *Pair Property* P2. We can associate $x_1 = 1$ with $x_2 = 6$ and put pair $(1,6)$ on path P_{LL} of $\overset{\triangle}{\triangledown}'_8$.

To determine a new pair, we consider the element $x_3 = mateD(x_2)$, in our example is $x_3 = 4$. This implies that this new pair will be associated to either path P_{LR} or P_{RR} because the choice of x_3 imposes the use of right edge on bottom level of $\overset{\triangle}{\triangledown}'_8$. To minimize edge conflicts, we consider, as second element of the pair, $x_4 = mateD(x_1)$ and we check if *Pair Properties* are respected. In our example this choice is in contradiction with property P1. Then we consider next element in sequence $DOWN_L$, that is $x_4 = 5$, and we obtain the valid pair $(4,5)$. We can not associate $(4,5)$ to path P_{LR} because a conflict on the upper level of $\overset{\triangle}{\triangledown}'_8$ would arise, then the path for pair $(4,5)$ is P_{RR}.

Following the same reasoning we produce the third pair and, obviously, the fourth is automatically given.

We consider $x_5 = mateU(x_4) = 6$, but this element is already used. Then we consider elements in the UP_R sequence until we find a valid element, that is $x_5 = 7$ in our example. We associate it with $x_6 = mateU(x_3) = 3$. We obtain the valid pair $(7,3)$. We can associate it both to path P_{LR} and to path P_{RL} because no conflict arises. Let P_{LR} be the chosen path. The last pair is $(8,2)$ and it must be put on path P_{RL}.

B. Algorithm for routing on $\overset{\triangle}{\triangledown}'_8$ factor

Given the $\overset{\triangle}{\triangledown}'_8$ factor and a permutation Π, here follows the algorithm to find the four pairs that will be used as inputs of the middle stage nodes of the considered MIN.

ROUTING ON $\overset{\vartriangle}{\triangledown}{}'_8$ ALGORITHM

`Input:`
$UP_L = [(1,2),(3,4)]$, $UP_R = [(5,6),(7,8)]$ fixed sequences
$DOWN_L = [(\pi(1),\pi(2))(\pi(5),\pi(6))]$, $DOWN_R = [(\pi(3),\pi(4))\ (\pi(7),\pi(8))]$ permutation sequences

`Output:`
for each input element, one of the paths P_{LL} P_{LR} P_{RL} P_{RR}

Step 1: Choice of the starting element

- if $UP_L \cap DOWN_L$ consists only of one element, then let it be the starting element x_1
- if $UP_L \cap DOWN_L$ consists of three elements, then let the lacking element of UP_L be the starting element x_1
- if $UP_L \cap DOWN_L$ consists of two elements **and** $UP \cap DOWN$ consists of one and only one pair p' then
  <u>*Procedure Check Permutation*</u>:
  if $p' \in UP_i$ with $i \in L, R$, then let x_1 be the element such that $(x_1 \in UP_i) \wedge (x_1 \notin p')$
- if previous cases are not verified then let $x_1 = 1$ be the starting element

Step 2: Determining the four pairs

1) associate x_1
   with an element $x_2 \in UP$ such that pair (x_1, x_2) respects properties P1 and P2; if x_1 has been obtained by *Procedure Check Permutation* then must be $x_2 \neq mateU(mateD(x_1))$
2) consider element $x_3 = mateD(x_2)$ in sequence $DOWN$ and element $x_4 = mateD(x_1)$ in sequence $DOWN$, check if pair (x_3, x_4) respects *Pair Properties* P1 and P2; if not, find an element x_4 in the same sequence $DOWN_i$, $i \in L, R$, containing $mateD(x_1)$, such that (x_3, x_4) satisfies *Pair Properties* P1 and P2
3) consider element $x_5 = mateU(x_4)$ in sequence UP, if it has been already put in a pair, find an element in sequence UP not considered yet; consider $x_6 = mateU(x_3)$ as second element if possible, otherwise find an element in sequence UP not considered yet, such that pair (x_5, x_6) satisfies *Pair Property* P1
4) form the last pair by using the two elements not considered yet

Step 3: Association of pairs to paths

1) associate
   pair (x_1, x_2) to path P_{LL}
2) associate pair (x_3, x_4) to path P_{LR}, if an edge conflict on first level of $\overset{\vartriangle}{\triangledown}$ arises, then associate (x_3, x_4) to path P_{RR}
3) choose the path for (x_5, x_6) in the following way:
   - if in previous step P_{LR} has been used, associate pair (x_5, x_6) to path P_{RL}, if an edge conflict on fourth level of $\overset{\vartriangle}{\triangledown}$ arises, then use path P_{RR}
   - if in previous step P_{RR} has been used, associate pair (x_5, x_6) to path P_{LR}, if an edge conflict either on first or fourth level of $\overset{\vartriangle}{\triangledown}$ arises, then use path P_{RL}
4) if path P_{LR} has not already been used in a previous step, then associate pair (x_7, x_8) to path P_{LR}, otherwise associate pair (x_7, x_8) to path P_{RL}

C. On rearrangeability of MIN equivalence classes

In this section we prove the correctness of algorithm ROUTING ON $\overset{\vartriangle}{\triangledown}{}'_8$ and the rearrangeability of equivalence classes of MINs that present the $\overset{\vartriangle}{\triangledown}{}'_8$ structure in their $\overset{\vartriangle}{\triangledown}$ decomposition factor.

Lemma 1. *Given a permutation of $N = 2^n$ elements, it is always possible to find $N/2$ pairs satisfying* Pair Properties *$P1$ and $P2$.*

Proof: Let us consider UP_i and $DOWN_j$, where $i, j \in L, R$ such that $UP_i \cap DOWN_j \neq 0$. Let x be an element belonging to $UP_i \cap DOWN_j$. To satisfy *Pair Properties* $P1$ and $P2$, x cannot be coupled neither with one of the other $N/2 - 1$ elements in UP_i, nor with one of the other $N/2 - 1$ elements in $DOWN_j$. In the worst case, $UP_i \cup DOWN_j = N - 1$, then there exists an element y belonging to $UP - (UP_i \cup DOWN_j)$ such that pair (x, y) satisfies $P1$ and $P2$. We can repeat this reasoning, after eliminating x and y from UP and $DOWN$.

Lemma 2. *Given a permutation of $N = 8$ elements, it is always possible to find a partition in pairs satisfying* Pair Properties *$P1$, $P2$ and $P3$.*

Proof: We show that after the choice of the first two pairs (x, y) and (w, z) satisfying $P1$, $P2$ and $P3$, we can always associate the remaining elements in pairs, still satisfying $P1$, $P2$ and $P3$. Usually, this initial choice can be arbitrary, but there are few cases in which it must be imposed to properly selected elements.

Let us consider UP_L, UP_R, $DOWN_L$ and $DOWN_R$ sequences as composed by *Mate Pairs*, MP, where a *Mate Pair* is an UP or $DOWN$ pair for which the element order is eliminated. In Figure 4 *Mate Pairs* are represented by means of circles. Let (x, y) be a pair satisfying properties $P1$ and $P2$, which existence is guaranteed by Lemma 1. Let (w, z) the pair obtained by taking $w = mateD(y)$ as first element, and choosing $mateD(x)$ or another element in the same $DOWN_i$, $i \in L, R$, as second element z. Lemma 1 guarantees that it is possible to determine an element z such that (w, z) satisfies properties $P1$ and $P2$. By construction, y and w belong to the same MP in $DOWN$. According to the arrangement of the four elements x, y, w, z with respect to the MPs in UP and $DOWN$ sequences, we have several different cases. Namely, we have three cases for the MPs in UP (see Figure 4):

- case 1 UP: x, y, w, z belong to 2 different MPs
- case 2 UP: x, y, w, z belong to 3 different MPs
- case 3 UP: x, y, w, z belong to 4 different MPs

and two cases for the MPs in $DOWN$ (see Figure 4):

- case a DOWN: x, y, w, z belong to 2 different MPs
- case b DOWN: x, y, w, z belong to 3 different MPs

All the possible relations are obtained by combining an UP and a $DOWN$ case together.

To prove the lemma, we must show that it is always possible to build the other two remaining pairs satisfying $P1$ and $P2$, and to associate the four pairs to the four paths P_{LL}, P_{LR}, P_{RL}, P_{RR} in such a way $P3$ is satisfied.

Because of the choice of the first two pairs, $DOWN$ sequences must assume one of the following configurations:

- case a: $DOWN_L((w, y), (v, s)), DOWN_R((x, z), (u, t)$
- case b: $DOWN_L((w, y), (v, s)), DOWN_R((x, u), (z, t))$

These configurations are not restrictive, because we are interested in the relationships among MPs in UP and $DOWN$ sequences.

Let us start to examine the possible cases:

Case 1 UP pairs (x, y) and (w, z) must be separated both on the first level and on the fourth, then the paths associated are P_{LL} and P_{RR}, respectively, see Figure 4; we distinguish two situations:

case a DOWN the remaining elements can be coupled in pairs, satisfying $P1$ and $P2$, that can be routed on P_{LR} and P_{RL} without conflicts, indifferently;

case b DOWN the remaining elements can be coupled in the two pairs (u, v) and (s, t), satisfying $P1$ e $P2$; in order to respect *Pair Property* P3, it suffices to associate (u, v) to path P_{LR} and (s, t)

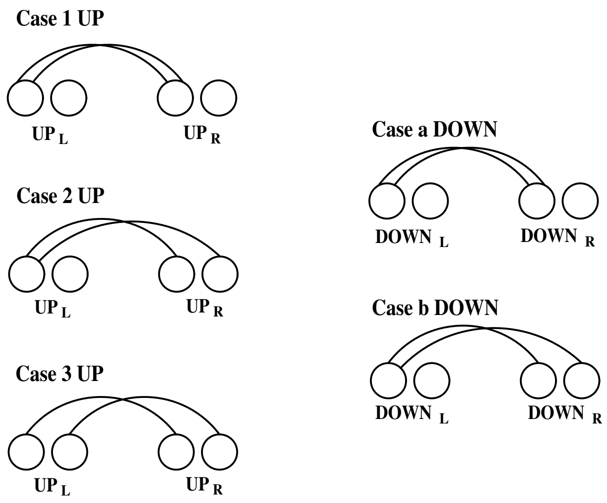

Fig. 4. The possible relations among elements of the two pairs are shown for the three UP cases and for the two $DOWN$ cases, on left and right side respectively. Since in *Mate Pairs* the element order is not relevant, they are represented simply as circles, without specifying element names.

to P_{RL} to guarantee that the elements in the same MP are separated on the fourth level (the two pairs can be routed indifferently on paths of the first level).

Case 2 UP this case imposes that the two pairs (x, y) and (w, z) chosen are associated to paths P_{LL} and P_{RR}, respectively, see Figure 4;

case a DOWN this case is similar to $case1.b$, where the first and fourth level are swapped, then we can resolve it in the same way, considering, wlog, x, t and z, u as the MP in UP_L;

case b DOWN this is the worst case. In fact, the particular configuration of elements can generate conflicts, then we need additional controls to avoid of building pairs satisfying P1 and P2, but not P3. To understand how to choose (x, y) and (w, z) in a correct way, observe that this configuration implies that two elements are in the same MP both in UP and in $DOWN$ sequences. We can distinguish the following cases, obtained by using the number of elements in $UP_L \cap DOWN_L$ (that is the same of $|UP_R \cap DOWN_R|$):

- $|UP_L \cap DOWN_L| = |UP_R \cap DOWN_R| = 1$ or $|UP_L \cap DOWN_L| = |UP_R \cap DOWN_R| = 3$ In this case, there is an *obligated pair*: we build the first pair by using the only element in the two intersections (if intersection is 1), or the remaining element with respect to the two intersections (if intersection is 3); the second pair is built as usual. This case is treated as case $2.a$.
- $|UP_L \cap DOWN_L| = |UP_R \cap DOWN_R| = 2$ Let N_{MP} be the number of MPs in $DOWN$ sequences that are equal to MP in UP sequences. By construction we have two cases:
  - $N_{MP} = 1$ Choosing (x, y) and associating it to path P_{LL}, it comes out that w, z and $t = mateD(x) = mateU(y)$ must be associated to path P_{RR}, generating a conflict. Note that $y = mateU(mateD(x))$. In this case, we must enforce that an element $y \neq mateU(mateD(x))$ is chosen. Then we can build the three remaining pairs without conflicts.
  - $N_{MP} = 0$ The assumptions imply that only one configuration is possible, namely $UP_L = ((x, w), (s, t))$, $UP_R = ((y, v), (z, u))$, $DOWN_L = ((w, y), (v, s))$ and $DOWN_R = ((x, u), (z, t))$, where u, s, t, v are not fixed. An ad-hoc solution for this situation is to associate pair (s, u) to path P_{LR}, and pair (t, v) to path P_{RL}.

- $|UP_L \cap DOWN_L| = 0 = |UP_R \cap DOWN_R|$ or $|UP_L \cap DOWN_L| = 4 = |UP_R \cap DOWN_R|$
  This case is not possible, due to definition of case $2.b$.

Case 3 UP we can associate pair (x, y) to path P_{LL} and pair (w, z) to path P_{LR}, respecting property $P3$;

case a DOWN the four remaining elements can be coupled into two pairs, respecting properties $P1$ and $P2$, and can be associated to P_{RL} and to P_{RR}, respecting property $P3$;

case b DOWN the remaining elements can be coupled in the two pairs (u, v) and (s, t), satisfying $P1$ and $P2$; in order to respect $P3$ property, it suffices to associate (u, v) to path P_{RR} and (s,t) to P_{RL}.

We have shown that there exists an assignment of pairs to paths satisfying P3 in all the possible cases.

Theorem 3. *Algorithm* ROUTING ON $\overset{\vartriangle}{\triangledown}{}'_8$ *produces pairs satisfying the three* Pair properties *P1, P2 and P3.*

Proof: The algorithm ROUTING ON $\overset{\vartriangle}{\triangledown}{}'_8$ provides a sequence of pairs, namely the algorithm builds pairs first, then it associates pairs to paths of $\overset{\vartriangle}{\triangledown}{}'_8$ giving the sequence. The correctness of the algorithm derives from Lemma 1 and Lemma 2.

Theorem 4. *All MINs in the class represented by the concatenation Butterfly-Butterfly, with $N = 8$ inputs, that is MINs having the $\overset{\vartriangle}{\triangledown}{}'_8$ factor, are rearrangeable.*

Proof: The sequence of pairs obtained by means of algorithm ROUTING ON $\overset{\vartriangle}{\triangledown}{}'_8$ represents the intermediate permutation on the MIN, that is the permutation presented at the inputs of the middle node stage. To realize this permutation it is possible to use the self-routing capability of the first $\log N - 1$ stages of the MIN, that is the routing based on the binary representation of the destinations. By using again the self-routing capability on the last $\log N$ stages, where inputs are the permuted elements obtained by the algorithm and outputs are given by the given permutation, the routing is completed.

Corollary 5. *All $(2 \log N - 1)$ stage MINs obtained by concatenating two logN stage MINs with $N = 8$ inputs are rearrangeable.*

Proof: As proved in [5], all $(2 \log N - 1)$ stage MINs obtained by concatenating two $\log N$ stage MINs with $N = 8$ inputs can be partitioned into two classes: the class represented by the concatenation of a Reverse Butterfly and a Butterfly, and the class represented by the concatenation of two Butterfly (or by two Reverse Butterfly).

MINs in the first class are equivalent to the Beneš, that is well known to be rearrangeable. An algorithm for setting the switches of a MIN in this class, exploiting its decomposition as $\overset{\triangledown}{\vartriangle}_8 \otimes \overset{\vartriangle}{\triangledown}_8$ is described in [6].

MINs in the second class can be decomposed as $\overset{\triangledown}{\vartriangle}_8 \otimes \overset{\vartriangle}{\triangledown}{}'_8$ and the above algorithm ROUTING ON $\overset{\vartriangle}{\triangledown}{}'_8$ allows to obtain the switch setting of such a MIN.

Theorem 6. *All MINs obtained by the concatenation of a Reverse Butterfly and a Butterfly whose first and second stages are swapped are rearrangeable and the permutation routing algorithm requires $O(N \log N)$ time.*

Proof: All MINs obtained by the concatenation of a Reverse Butterfly and a Butterfly whose first and second stages are swapped, can be decomposed as $\overset{\triangledown}{\vartriangle} \otimes \overset{\vartriangle}{\triangledown}$ with the $\overset{\vartriangle}{\triangledown}$ factor having $\overset{\vartriangle}{\triangledown}{}'_8$s in the inner stages (see [5]).

Then it is sufficient to show that for the $\overset{\vartriangle}{\triangledown}$ factor, we can route inputs to first level of the $\overset{\vartriangle}{\triangledown}{}'_8$ structures and outputs to the last level of $\overset{\vartriangle}{\triangledown}'_8$ structures, in such a way we have the same groups of elements as inputs and outputs for each $\overset{\vartriangle}{\triangledown}'_8$.

Let $n = \log N$. The proof is by induction on n. The base case corresponds to $\overset{\vartriangle}{\triangledown}'_8$, for which algorithm ROUTING ON $\overset{\vartriangle}{\triangledown}'_8$ is given, that is $n = 3$. Hence assume that the result is true for a MIN with $N/2$ inputs and prove for a MIN with N inputs. The key idea is to observe that eliminating the two roots of $\overset{\vartriangle}{\triangledown}$ we obtain two smaller $\overset{\vartriangle}{\triangledown}$s. Hence, it suffices to decide for each input whether it is to be routed through the left or the right sub-$\overset{\vartriangle}{\triangledown}$, and do the same for the outputs. The only constraints that we must satisfy when we choose the left or the right sub-$\overset{\vartriangle}{\triangledown}$, both upward and downward, are: i) elements forming an input pair must go to different sub-$\overset{\vartriangle}{\triangledown}$s from the root of Δ, ii) elements forming an output pair must go to different sub-$\overset{\vartriangle}{\triangledown}$s from the root of ∇.

This can be obtained by proceeding as in the Looping algorithm, that is we start by any input and we choose the sub-$\overset{\vartriangle}{\triangledown}$, we associate the mate of this element in the outputs to the other sub-$\overset{\vartriangle}{\triangledown}$, then we consider the mate in the inputs of this new element and proceed in this way until the loop is closed. If we do not have considered all the inputs, we choose any new starting element and repeat the procedure until all elements are associated either to the left sub-$\overset{\vartriangle}{\triangledown}$ or to the right one.

Each sub-$\overset{\vartriangle}{\triangledown}$ can be handled by the inductive hypothesis and therefore the correctness is proved.

About the time complexity, we consider each input or output element once, for each of these elements we choose the left or right sub-$\overset{\vartriangle}{\triangledown}$. This choice is repeated $O(\log N)$ times due to the structure of $\overset{\vartriangle}{\triangledown}$. Hence, the global time complexity is $O(N \log N)$.

Remark 1. *The proof of rearrangeability due to Raghavendra and Varma [14] for a network in the MIN class identified as $\overset{\triangledown}{\vartriangle}_8 \otimes \overset{\vartriangle}{\triangledown}'_8$, namely the Five-Stage Shuffle/Exchange Network for $N = 8$, is obtained by means of an algorithm providing the switch setting for any permutation. This algorithm is applicable only to the network for which it is designed, since it exploits the network topology. On the contrary algorithm* ROUTING ON $\overset{\vartriangle}{\triangledown}'_8$ *is valid for any MIN in the class. Furthermore, an analysis of the two algorithms allows to verify that the new proposed algorithm is more efficient. In fact, taking into account the comparison operation, that is the more frequent and expensive operation for both algorithm, we have that* ROUTING ON $\overset{\vartriangle}{\triangledown}'_8$ *Algorithm requires a minor number of comparisons.*

IV. CONCLUSIONS AND FUTURE WORK

In this paper we have provided an algorithm to realize any permutation Π on MINs belonging to the complementary equivalence class of Beneš network with $N = 8$ inputs, with respect to the decomposition as $\overset{\triangledown}{\vartriangle} \otimes \overset{\vartriangle}{\triangledown}$.

By means of this algorithm, we give a constructive proof of rearrangeability for the equivalence class of networks with N inputs, represented by the concatenation of a Reverse Butterfly and a Butterfly, whose first and second stages are reversed; for this class the rearrangeability was not known. The time complexity is $O(N \log N)$ which is the same as the well-known Looping algorithm for the Beneš network. Notice that lower values for the time complexity have been obtained only for special class of permutations.

The interest of the LCP based decomposition approach is that: i) it is possible to study the routing exploiting only the $\overset{\vartriangle}{\triangledown}$ factor, that is a simpler structure than the considered MIN, ii) proving the rearrangeability of specific network by using its $\overset{\vartriangle}{\triangledown}$ factor immediately implies the rearrangeability of the whole equivalence class, iii) more general algorithms are obtained by means of decomposition, since they are not tied to the network topology.

Finally, a deep understanding of the features of a MIN, provided by the utilization of its LCP based factors, can lead to the proof of rearrangeability of other interesting classes of networks, e.g. the class containing the Omega-Omega, that is an open problem since a long time.

REFERENCES

[1] H. Bao, F. K. Hwang and Q .Li: Rearrangeability of bit permutation networks, *Theoretical Computer Science* vol. 352, pp. 197–214, 2006.

[2] V. E. Beneš: On Rearrangeable Three-Stage Connecting Networks, *Bell Syst. Tech. J.*, XLI, pp.1481-1492, 1962.

[3] V. E. Beneš: Permutation Groups, Complexes, and Rearrangeable Connecting Networks, *Bell Syst. Tech. J.*, vol 43, pp.1619-1640, 1964.

[4] J. C. Bermond, J. M. Fourneau and A. Jean-Marie, "Equivalence of Multistage Interconnection Networks" *Inform. Proc. Letters*, 26, pp. 45-50, 1987.

[5] T. Calamoneri and A. Massini: Efficiently Checking the Equivalence of Multistage Interconnection Networks, *Journal of Parallel and Distributed Computing*, 64, pag. 135-150, 2004.

[6] T. Calamoneri and A. Massini: A new approach to the rearrangeability of $2logN - 1$ stage MINs, *Proc. IASTED Internat. Symp. Applied Informatics (AI 2001)*, pp. 365-370, 2001.

[7] N. Das: More on Rearrangeability of Combined $(2n - 1)$-stage Networks, *Journal of Systems Architecture*, pp. 207-222, 2005.

[8] S. Even and A. Litman: Layered Cross Product - A technique to construct interconnection networks. 4^{th} *ACM SPAA*, pp. 60-69, 1992.

[9] N. Golbandi and A. Litman: Characterizations of Generalized Butterfly Networks, *Technical Report CS 2004-10*, Technion Computer Science Department, 2004.

[10] Q. Hu, X. Shen and J. Yang: Topologies of Combined $(2 \log N - 1)$-stage Interconnection Networks, *IEEE Trans. Comput.*,pp. 118-124, 1997.

[11] K. Y. Lee: On the Rearrangeability of $(2 \log N - 1)$-Stage Permutation Networks, *IEEE Trans. Comput.*, 46(1), C34, pp. 412-425, 1985.

[12] K. Y. Lee: A New Beneš Networks Control Algorithm, *IEEE Trans. Comput.*, C36, pp. 768-772, 1987.

[13] A. Paz: A Theory of Decomposition into Prime Factors of Layered Interconnection Networks, *Technical Report CS 2001-20*, Technion Computer Science Department, 2001.

[14] C. S. Raghavendra and A. Varma,: Rearrangeability of The Five-Stage Shuffle/Exchange Network for $N = 8$, *IEEE Trans. Commun.*, COM-356, pp 808-812, 1987.

[15] A. Varma and C. S. Raghavendra: Rearrangeability of Multistage Shuffle/Exchange Networks, *IEEE Trans. Comput.*, 36(10) pp 1138-1143, 1988.