Generalized Rearrangeable Networks with Recursive Decomposition Structure

Myung-Kyun Kim, Hyunsoo Yoon, and Seung-Ryoul Maeng

Abstract

This paper proposes a class of rearrangeable networks, called generalized rearrangeable networks(GRNs). GRNs are obtained from the Benes network by rearranging the connections between stages and the switches within each stage. The GRNs constitute all of the rearrangeable networks which have the recursive decomposition structure and can be routed by the outside-in decomposition of permutations as the Benes network. This paper also presents a necessary condition for a network to be a GRN and a network labeling scheme to check if a network satisfies the condition. The general routing algorithm for the GRNs is given by modifying slightly the looping algorithm of the Benes network.

I. Introduction

Multistage interconnection networks(MINs) have been widely used in multiprocessor systems to connect thousands of processors and memory modules. A MIN is called a rearrangeable network if all permutations can be realized by one pass through the network. A number of researches for rearrangeable networks and their routing schemes have been done for many years[4, 5, 6, 12]. An $N\times N$ MIN can be constructed into k stages, each of which consists of (N/a) $a\times a$ switches. A MIN with $a\times a$ switches, where $a>2$, can be easily extended from a MIN with 2×2 switches, so MINs with 2×2 switches are only considered and $\log N$ indicates $\log_2 N$ in this paper. It was shown that the number of stages that is necessary for an $N\times N$ MIN to be rearrangeable is $2\log N-1$ [9]. Many researchers tried to find rearrangeable networks with $2\log N$(or $2\log N-1$) stages and routing algorithms for those networks. Benes network[4] is a rearrangeable network which is composed of 2×2 switches and many routing algorithms for the Benes network were proposed[3, 8, 11]. Lee[6] proved that an omega-omega^{-1} network is rearrangeable and suggested a routing algorithm for the network. Yeh and Feng[12] introduced a class of rearrangeable networks, called the equivalent Benes networks, which are topologically equivalent to the Benes network, and proposed a switch labeling scheme to show the topological equivalence relationships. Recently, Kim et. al.[5] introduced a class of rearrangeable networks, called symmetric BPMINs(Bit-Permute Multistage Interconnection Networks), which have the same recursive decomposition structures as the Benes network.

The rearrangeable networks such as the equivalent Benes network[12] and the symmetric BPMIN[5] can be obtained from the Benes network by rearranging the connections between stages and rearranging the switches within stages. Both of the above rearrangeable networks contain only the networks which have buddy property[1] between the switches of adjacent stages. This paper further generalizes the rearrangeable networks and proposes a class of rearrangeable networks, called generalized rearrangeable networks(GRNs). The GRNs can be obtained from the Benes network by rearranging the inter-stage connections and rearranging the switches within stages. GRNs have the same recursive decomposition structure as the Benes network and are controllable by the outside-in decomposition of permutations. It is shown that GRNs consist of all of the rearrangeable networks with recursive decomposition structure. This paper also presents a necessary condition for a network to satisfy to be a GRN and a network labeling scheme to check if a network satisfies the condition. A general routing algorithm for the GRNs is given by modifying slightly the looping algorithm of the Benes network.

Section 2 describes a few preliminary definitions and lemmas on permutations and defines the GRNs. Section 3 describes a necessary condition for a network to satisfy to be a GRN and a network labeling scheme to check the condition. A general routing algorithm for the GRNs is also described in this section. Section 4 describes the conclusion.

II. Generalized Rearrangeable Networks

In an $N\times N$ MIN, where $N=2^n$, the stages are numbered 0, 1, $\cdots$, $2\log N$-2 from left to right and the switches of each stage are

Manuscript received March 24, 1997; accepted July 31, 1997.
M. K. Kim is with Computer Engineering Dept., Woosuk University.
H. Yoon and S. R. Maeng are with Computer Science Dept., KAIST.

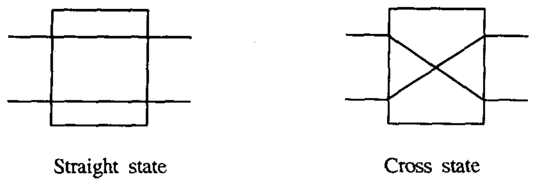


Fig. 1. Two possible states of a switch.

numbered 0, 1, ···, $N/2$-1 from top to bottom. Each switch has two possible states: the straight state and the cross state as shown in Fig. 1. The following describes some backgrounds about permutations and then discusses the rearrangeability of MINs by their permutations.

Definition 2.1 Let $S = \{0, 1, \cdots, N\text{-}1\}$. A permutation of a set S is a bijection from S onto S. The set of permutations on S is denoted by G_S.

G_S is a symmetric group under the composition of permutations and has $N!$ elements[7]. A network Ψ is called rearrangeable if and only if every permutation P in the symmetric group G_S is passable on the network. Let α and β be two networks and let $I = \alpha \cdot \beta$ be a network obtained by concatenating α and β. Every permutation P passable on the network I can be described as a product of two permutations P_α and P_β such that $P = P_\alpha \cdot P_\beta = P_\beta(P_\alpha)$, where P_α is the permutation that corresponds to the setting of the network α and P_β to that of the network β. Thus, the set of permutations passable on the network obtained by concatenating two networks are described as follows.

Definition 2.2 Let Γ_α be the set of all permutations passable on network α and Γ_β be the set of all permutations passable on network β. The set of permutations passable on network $I = \alpha \cdot \beta$ is defined as follows:

$$\Gamma_\gamma = \{ P = P_\alpha \cdot P_\beta \mid P_\alpha \in \Gamma_\alpha \text{ and } P_\beta \in \Gamma_\beta \}.$$

Lemma 1 Let Ψ be a rearrangeable network, and α be a network generating a fixed permutation P_α and β be a network generating a fixed permutation P_β, then $\Phi = \alpha \cdot \Psi \cdot \beta$, which is a serial concatenation of the networks α, Ψ, and β, is also rearrangeable.

Proof Let Γ_Ψ be a set of permutations passable on the network Ψ, then $\Gamma_\Psi = G_S$ from Lemma 1. Let P be an arbitrary permutation in G_S. If P is passable on the network Φ, then the network Φ is rearrangeable. Because Ψ is rearrangeable, it can route the permutation ($P_\alpha^{-1} \cdot P \cdot P_\beta^{-1}$) where P_α^{-1} and P_β^{-1} are the inverses of the permutations P_α and P_β, respectively. If the switches of the network Ψ are set to route the permutation ($P_\alpha^{-1} \cdot P \cdot P_\beta^{-1}$), then the network Φ comes to route the permutation $P_\alpha \cdot (P_\alpha^{-1} \cdot P \cdot P_\beta^{-1}) \cdot P_\beta = P$. Thus, the network Φ can route all of the permutations in G_S, so Φ is rearrangeable. □

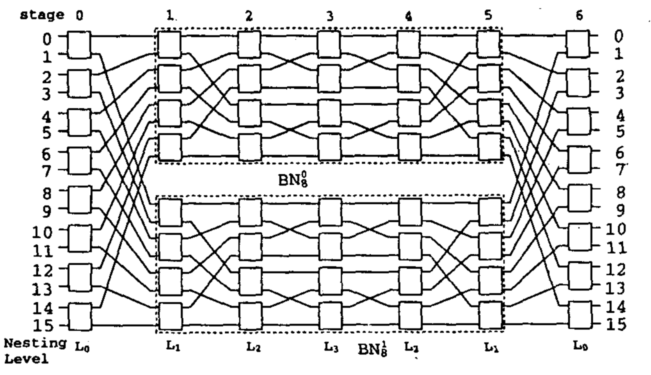


Fig. 2. 16×16 Benes network, BN_{16}, with recursive decomposition structure.

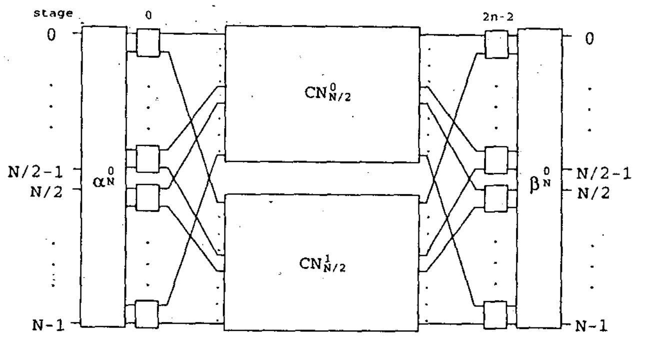


Fig. 3. The network structure of a CRN CN_N.

The Benes network is known as a rearrangeable network with recursive decomposition structure. Fig. 2 depicts the recursive decomposition structure of $N \times N$ ($N = 16$) Benes network, BN_N. By removing the switches and their incident connections of stage 0 and $2n$-2, the network BN_N is decomposed into two disjoint sub-networks $BN^0_{2^{n-1}}$ and $BN^1_{2^{n-1}}$, each of which is $(N/2)\times(N/2)$ Benes network and the two links of each switch in nesting level L_0 are connected to different sub-networks of nesting level L_1. Similarly, the sub-network $BN^0_{2^{n-1}}$($BN^1_{2^{n-1}}$) can also be decomposed into two disjoint sub-networks $BN^0_{2^{n-2}}$ and $BN^1_{2^{n-2}}$($BN^2_{2^{n-2}}$ and $BN^3_{2^{n-2}}$). In general, by removing the switches and their incident connections of stages from 0 to i-1 and from $2n$-2 to $2n$-i-1 where $1 \le i \le n\text{-}1$, the network BN_N is decomposed into 2^i disjoint sub-networks such as $BN^0_{2^{n-i}}$, $BN^1_{2^{n-i}}$, ···, $BN^{2^i-1}_{2^{n-i}}$, each of which is $2^{n-i} \times 2^{n-i}$ Benes network. The two links of a switch in nesting level L_i, $0 \le i \le n\text{-}2$, are connected to different sub-networks of the next nesting level L_{i+1}.

Using Lemma 1, the $N \times N$ Benes network, BN_N, can be generalized to an $N \times N$ canonical rearrangeable networks(CRNs), CN_N, by substituting each $BN^k_{2^m}$ with $CN^k_{2^m}$ such that $CN^k_{2^m} = \alpha^k_{2^m} \cdot BN^k_{2^m} \cdot \beta^k_{2^m}$ where $\alpha^k_{2^m}$ and $\beta^k_{2^m}$ are arbitrary $2^m \times 2^m$ connections and $1 \le m \le n$ and $0 \le k \le 2^{n-m}\text{-}1$. Fig. 3 depicts the network structure of an $N \times N$ CRN, CN_N. The CN_N also has the same recursive decomposition structure as the Benes network. By removing the switches of stages from 0 to i-1 and from $2n$-i-1 to $2n$-2 where $1 \leq i \leq n$-1, the network CN_N is decomposed into 2^i disjoint sub-networks such as $CN^0_{2^{n-i}}$, $CN^1_{2^{n-i}}$, $\cdots$, $CN^{2^i-1}_{2^{n-i}}$, each of which is a $2^{n-i} \times 2^{n-i}$ CRN. The two links of each switch at nesting level L_i are connected to different sub-networks of the next nesting level L_{i+1}. The CRNs can further be generalized by rearranging the switches within stages. Thus, we can define the $N \times N$ generalized rearrangeable networks as follows.

Definition 2.3 The class of networks obtained from the $N \times N$ Benes network, BN_N, by the following construction procedure is called an $N \times N$ generalized rearrangeable network(GRN) and denoted by GN_N:

1. substitute each sub-network $BN^k_{2^m}$ with $CN^k_{2^m}$ such tha $CN^k_{2^m}$t $= \alpha^k_{2^m} \cdot BN^k_{2^m} \cdot \beta^k_{2^m}$ where $\alpha^k_{2^m}$ and $\beta^k_{2^m}$ are arbitrary $2^m \times 2^m$ connections and $1 \leq m \leq n$ and $0 \leq k \leq 2^{n-m}$-1, and
2. rearrange the switches within stages.

Theorem 1 All of the $N \times N$ GRNs are rearrangeable.

Proof From Lemma 1 and the above discussions on the construction of GRNs, it can be easily seen that the theorem is true. □

Theorem 2 The GRNs consist of all of the rearrangeable networks with recursive decomposition structure.

Proof From the above construction procedure of the $N \times N$ GRNs, it can be easily seen that the GRNs consist of all of the rearrangeable networks with recursive decomposition structure, where the two ports of each switch at nesting level L_i are connected to different sub-networks of the next nesting level L_{i+1}. Thus, the theorem is true if it can be shown that the networks, where the two ports of a switch at nesting level L_i are connected to the same sub-networks of the next nesting level L_{i+1}, are not rearrangeable. Since a GRN is obtained from a CRN by a simple rearrangement of the switches within stages, if it is shown to be true for the CRNs, then the same is also true for the GRNs. Let CN_N be a CRN obtained from a GRN CN_N by the rearrangement of the switches within stages. Let's assume that the two ports of a switch, say X, of stage 0 of the CN_N are connected to the same sub-network $CN^0_{N/2}$ of the next nesting level 1 as shown in Figure 4. Let Y be a switch of stage $2n$-2 whose two ports are connected to different sub-networks of the next nesting level 1. Let i and j be the two input terminals of X, and k and l be two output terminals of Y. Since only one connection link exists between $CN^0_{N/2}$ and the switch Y, the permutations, where the destinations of i and j are the output terminals of the switch Y, can not be routed in one pass. Thus, the CRN CN_N is not rearrangeable, and neither the GRN GN_N. From the above fact, the theorem is true. □

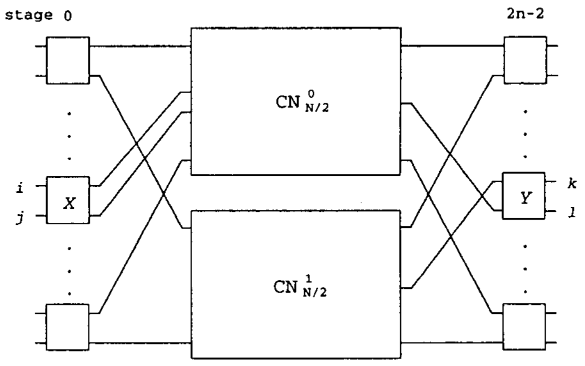
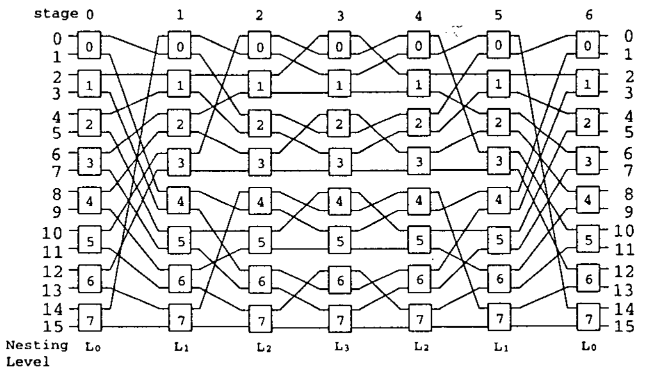
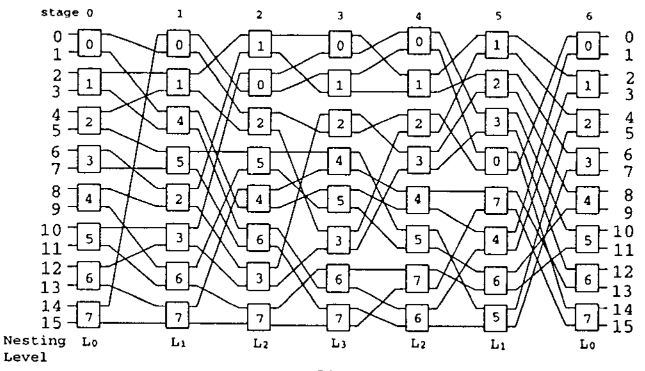


Fig. 4. An example of Theorem 2.



(a)



(b)

Fig. 5. Examples of the CRN and GRN. (a) a CRN. (b) a GRN obtained from (a).

Lemma 2 Both of the symmetric BPMINs [5] and the equivalent Benes networks [12] are included in the GRNs.

Proof The symmetric BPMINs and equivalent Benes networks have the same recursive decomposition structure as the GRNs, thus the theorem is true. □

Fig. 5 depicts examples of a GRN and a CRN which is not a symmetric BPMIN. Fig. 5(b) is a GRN obtained from the CRN of Fig. 5(a). The numbers in each switch represents the mapping relationships between the two networks. The first half of the network in Fig. 5(a) is a 16×16 zeta network nonequivalent to the baseline network [2], and the network of the last n-1 stages is the reverse of the first n-1 stages of the zeta network. Thus, the GRNs include the rearrangeable networks constructed from the nonequivalent MINs as shown in Fig. 5.

III. Network Labeling Scheme and Routing Algorithm of GRNs

1. Network Labeling Scheme of GRNs

Before the description of the labeling scheme, we suggest a necessary condition for a network to be a GRN. The network labeling scheme checks if a network satisfies the condition, and labels the switches and ports of the network according to the sub-network structure if the labeling succeeds. Since a GRN is obtained from a CRN by simply rearranging the switches within stages, the network labeling scheme of a GRN is the same as that of a CRN. Thus, for the sake of easy description, we will describe a necessary condition and a network labeling scheme for a CRN. From the network decomposition structure as shown in Fig. 3, an $N \times N$ CRN, CN_N, satisfies the following conditions:

(C1)' $CN^0_{N/2}$ and $CN^1_{N/2}$ are $(N/2) \times (N/2)$ CRNs independent with each other,

(C2)' the number of switches of stage 0(and $2n$-2), which are connected to the sub-networks $CN^0_{N/2}$ and $CN^1_{N/2}$, is $(N/2)$, and

(C3)' the two ports of each switch of stage 0(and $2n$-2) are connected to different sub-networks of stage 1(and $2n$-3).

In general, there are 2^i sub-networks at nesting level L_i in CN_N such as $CN^0_{2^{n-i}}$, $CN^1_{2^{n-i}}$, $\cdots$, $CN^{2^i-1}_{2^{n-i}}$. Each sub-network of nesting level L_i, $CN^j_{2^{n-i}}$ where $0 \le j \le 2^i$-1, has the same recursive decomposition structure as the network CN_N, and is divided into two disjoint sub-networks, $CN^{2j}_{2^{n-i-1}}$ and $CN^{2j+1}_{2^{n-i-1}}$, at the next nesting level L_{i+1}. Therefore, in a CRN CN_N, all of the sub-networks of nesting level L_i, $0 \le i \le n$-2, satisfy the same conditions as the above, thus those conditions can be generalized as follows. An $N \times N$ CRN CN_N satisfies the following three conditions, for all i and j where $0 \le i \le n$-2 and $0 \le j \le 2^i$-1,

(C1) each sub-network of nesting level L_i, $CN^j_{2^{n-i}}$, is divided into two disjoint sub-networks, $CN^{2j}_{2^{n-i-1}}$ and $CN^{2j+1}_{2^{n-i-1}}$, at the next nesting level L_{i+1},

(C2) the number of switches of stage i(and $2n$-i-2), which are connected to the two sub-networks of nesting level L_{i+1}, $CN^{2j}_{2^{n-i-1}}$ and $CN^{2j+1}_{2^{n-i-1}}$, is 2^{n-i-1}, and

(C3) the two ports of each switch of stage i($CN^{2j}_{2^{n-i-1}}$ and $2n$-i-2) in the sub-network $CN^j_{2^{n-i}}$ are connected to two different sub-networks, and $CN^{2j+1}_{2^{n-i-1}}$, from each other.

If the CRN and CN are replaced by GRN and GN in the above conditions, (C1), (C2), and (C3), they become the necessary conditions for a GRN.

The network labeling scheme checks whether the network satisfies the above three conditions (C1), (C2), and (C3), and labels the switches of the network, the output ports of the switches at stages from 0 to n-2, and the input ports of the switches at stages from n to $2n$-2 if the labeling succeeds. The switches of nesting level L_0 are labeled as NIL(empty string), and the switches of the nesting level L_i in the sub-network $GN^j_{2^{n-i}}$ are labeled as an i bit string $b_{i-1}\, b_{i-2} \cdots b_0 = j$, where b_k, $0 \le k \le i$-1, is 0 or 1. The output ports of the switches of stage i(and the input ports of the switches of stage $2n$-i-2) in the sub-network $GN^j_{2^{n-i}}$ are labeled as 0 if they are connected to the sub-network of nesting level L_{i+1}, $GN^{2j}_{2^{n-i-1}}$, and labeled as 1 if they are connected to the sub-network of nesting level L_{i+1}, $GN^{2j+1}_{2^{n-i-1}}$. The labeling of the sub-network of nesting level L_i, $GN^j_{2^{n-i}}$, is done recursively by first labeling the sub-networks of nesting level L_{i+1}, $GN^{2j}_{2^{n-i-1}}$ and $GN^{2j+1}_{2^{n-i-1}}$, and next labeling the switches and their ports of nesting level L_i in the sub-network $GN^j_{2^{n-i}}$. The network labeling procedure, Label_GRN, is described in Fig. 6.

The procedure Label_GRN(GN, i, j, SW, T) labels the switches and their ports of the jth sub-network of nesting level L_i of the network GN_N, $GN^j_{2^{n-i}}$. SW is a switch in the sub-network $GN^j_{2^{n-i}}$ where the labeling begins, and T is an i bit string of size i to be labeled in the switch SW, where $T = b_{i-1}\, b_{i-2} \cdots b_0 = j$. Label_GRN(GN, i, j, SW, T) begins by first labeling the switch SW as T and next labeling the two sub-networks of nesting level L_{i+1},

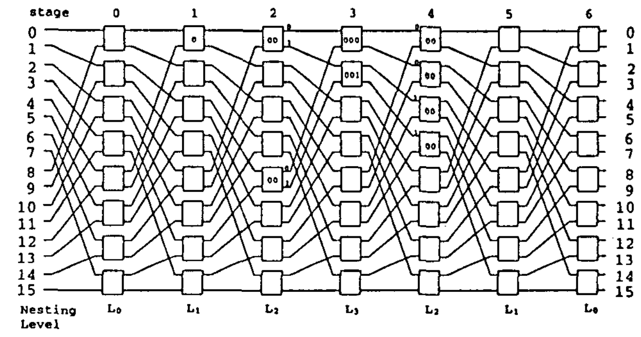
```
Procedure Label_GRN(GN, $i$, $j$, SW, T)
begin
    Label SW as T;
    if ($i$ = $n$-1) then return;
    Next_SW1 = the switch of stage $i+1$ connected with the upper
        link of SW;
    Label_GRN(GN, $i+1$, $2j$, Next_SW1, T · 0);
    Next_SW2 = the switch of stage $i+1$ connected with the lower
        link of SW;
    Label_GRN(GN, $i+1$, $2j+1$, Next_SW2, T · 1);
    if (Check_Labels($i+1$, $2j$, $2j+1$) = -1) then stop;  /* Violates (C1) */
    Propagate_Labels($i+1$, $2j$, $2j+1$);
    if (Number_of_SW($i$) $\neq 2^{n-i-1}$ OR Number_of_SW($2n$-$i$-2) $\neq 2^{n-i-1}$)
                then stop;              /* Violates (C2) */
    if (Check_Ports($i$) = -1) then stop;       /* Violates (C3) */
    Set_Labels($i$);
    return;                         /* Labeling done successfully. */
end
```

Fig. 6. Network labeling procedure for a GRN GN_N.

$GN^{2j}_{2^{n-i-1}}$ and $GN^{2j+1}_{2^{n-i-1}}$. The labeling of the sub-network $GN^{2j}_{2^{n-i-1}}$ starts from a switch of stage *i+1*, Next_SW1, which is connected with the upper output link of the switch SW, and labels the switch Next_SW1 as T · 0 = 2*j*. The labeling of the sub-network $GN^{2j+1}_{2^{n-i-1}}$ starts from a switch of stage *i*+1, Next_SW2, which is connected with the lower output link of the switch SW, and labels the switch Next_SW2 as T · 1 = 2*j*+1. After that, it checks the switches of stages *i*+1 and 2*n*-*i*-3 labeled by the labeling of the two sub-networks $GN^{2j}_{2^{n-i-1}}$ and $GN^{2j+1}_{2^{n-i-1}}$, and stops the labeling if those switches are doubly labeled, which means the violation of the condition (C1). If the labeling of the two sub-networks $GN^{2j}_{2^{n-i-1}}$ and $GN^{2j+1}_{2^{n-i-1}}$ is done successfully, then the switches of stages *i*+1 and 2*n*-*i*-3 in the sub-networks $GN^{j}_{2^{n-i}}$ and $GN^{2j+1}_{2^{n-i-1}}$ propagate their labels to the next outer stages *i* and 2*n*-*i*-2, respectively. Each of the switches receiving the labels at nesting level L_i labels itself as the most significant *i* bits of the received labels, and labels each of its two ports as the least significant bit of the label. To satisfy the conditions (C2) and (C3), the number of switches of stage *i*(or 2*n*-*i*-2) receiving the labels must be 2^{n-i-1} (condition (C2)), and the two labels of each switch received through the upper and lower ports must be different only in their least significant bits(condition (C3)). The procedure completes the labeling of the sub-network $GN^{j}_{2^{n-i}}$ successfully if those conditions are satisfied, and stops the labeling if not. Initially, the labeling starts by calling Label_GRN(GN, 0, 0, SE(0,0), NIL), where SE(0,0) denotes the switch of stage 0 connected with the input terminal 0.

Definition 3.1 The functions used in the network labeling procedure Label_GRN are the following.

- Check_Labels(*i*+1, 2*j*, 2*j*+1) is a function to check whether the switches of nesting level L_{i+1} in the sub-networks $GN^{2j}_{2^{n-i-1}}$ and $GN^{2j+1}_{2^{n-i-1}}$, are doubly labeled, and return -1 if so and 0 if not.
- Propagate_Labels(*i*+1, 2*j*, 2*j*+1) is a function to propagate the labels of the switches of nesting level L_{i+1} in the sub-networks $GN^{2j}_{2^{n-i-1}}$ and $GN^{2j+1}_{2^{n-i-1}}$ to the switches of the next nesting level L_i.
- Number_of_SW(*i*) is a function to return the number of switches of stage *i* which receive the labels propagated from the sub-networks $GN^{2j}_{2^{n-i-1}}$ and $GN^{2j+1}_{2^{n-i-1}}$.
- Check_Ports(*i*) is a function to check whether the two labels received by each switch of nesting level L_i are different only in their least significant bits, and returns -1 if it is not true.
- Set_Labels(*i*) labels each switch of nesting level L_i in the sub-network $GN^{j}_{2^{n-i}}$ as the most significant *i* bits of the received labels, and labels each of its two ports as the least significant bit of the label.


(a)

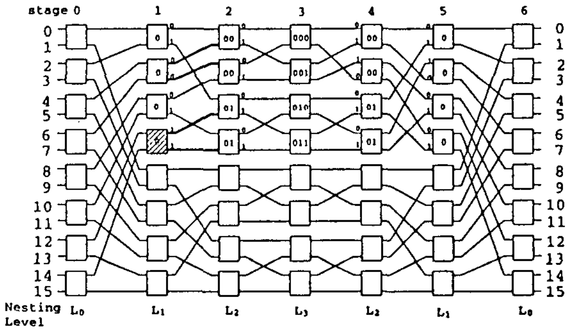

(b)

Fig. 7. Network labeling examples which violate the conditions. (a) 2log*N*-1 stage shuffle-exchange network which violates the condition (C2). (b) an example of networks which violate the condition (C3).

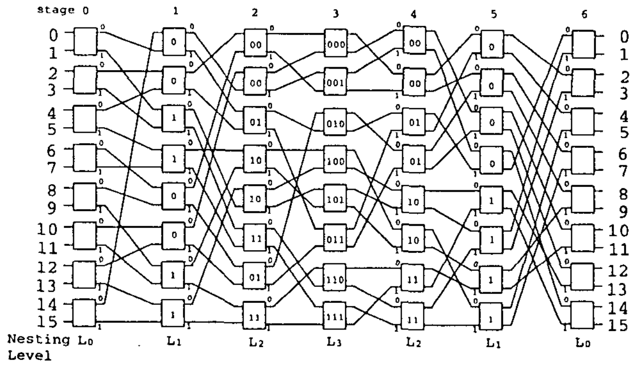


Fig. 8. Network labeling example of the GRN of Fig. 5(b).

Fig. 7 and Fig. 8 depicts the labeling examples for some networks. Fig. 7(a) depicts the network labeling result of 2log*N*-1 stage shuffle-exchange network which has no recursive decomposition structure. The number of switches of stage 4 receiving the labels from the two sub-networks of nesting level L_3, which are labeled as 000 and 001, respectively, must be 2 not 4, thus the labeling fails because it violates the condition (C2). Fig. 7(b) depicts the labeling example of network which violates the condition (C3). In the figure, the two output ports of the dotted switch are connected to the same sub-network of nesting level L_2, which is labeled as 00, and those of the hatched switch are also connected to the same sub-network of nesting level L_2, which is labeled as 01. Thus, the network labeling fails. Fig. 8 depicts a successful network labeling example for the GRN of Fig. 5(b).

2. *Routing Algorithm of GRNs*

A GRN can be labeled by the labeling procedure Label_GRN. Each switch keeps the labels of its two ports given by the labeling procedure. Let the ports of the switches in each stage be numbered 0 to N-1 from top to bottom, and let P be an arbitrary permutation to be routed on a GRN GN$_N$. The routing of P in GN$_N$ is done as follows. At first, the input terminal i, $0 \le i \le N$-1, is tagged as its destination $P(i)$, and the output terminal i is tagged as i. The routing procedure consists of n steps. At step i where $0 \le i \le n$-2, the input-side tag set is propagated to the input ports of the switches at stage i according to the connection between the stages i-1(the input terminals when $i = 0$) and i, and the output-side tag set is propagated to the output ports of stage $2n$-i-2 according to the connection between the stages $2n$-i-2 and $2n$-i-1(the output terminals when $i = 0$). Next, the switches of stage i and $2n$-i-2 are set by the modified algorithm of the looping algorithm of the Benes network. After that, the input-side and output-side tag sets are propagated to the ports of stage i+1 and the ports of stage $2n$-i-3, respectively, and it continues the next step i+1. After the step n-2, the same set of tags meet at each switch of stage n-1, and so, at stage n-1, each switch can be set by connecting the input and output ports with the same tags. Fig. 9 presents the routing algorithm for a GRN. The routing algorithm Route_GRN(GN$_N$, P) sets the switches of the GRN GN$_N$ to route the permutation P.

Definition 3.2 The following notations and function are used in the routing algorithm Route_GRN.

- The input side and output side tag sets are denoted by I and D. At the beginning of routing step i, I(or D) is tagged at the input(or output) ports of stage i(or $2n$-i-2) and $I(k)$(or $D(k)$) denotes the tag given to the input(or output) port k of stage i(or $2n$-i-2).
- The two ports in a switch are called buddies and buddy(x) denotes the buddy port of port x.
- The two input side and output side ports with the same tags are called peers, and peer(T), where T is an input(or output) side tag, indicates the peer port of the port with tag T.
- The control setting result is stored in a switch control matrix $M[0:2n$-$1, 0:N$-$1]$. When $0 \le s \le n$-1, $M[s, 0:N$-$1]$ denote the connection states of switches of stage s from their input ports to the output ports. If $M[s, t]$ is 0, the tth input port of stage s is connected to the output port with label 0, and, otherwise, the input port is connected to the output port with label 1. When $n \le s \le 2n$-2, $M[s, 0:N$-$1]$ denote the connection states of switches of stage s from their output ports to the input ports. If $M[s, t]$ is 0, the tth output port of stage s is connected to the input port with label 0, and, otherwise, the output port is connected to the input port with label 1. Initially, all elements of M are initialized as NIL(no value).
- Propagate_Tags(i) is a function to propagate the input(or output) tags at the input(or output) ports of stage i($2n$-i-2) to the switches of the next inner stage i+1(and $2n$-i-3) through the switches of the stage i and $2n$-i-2) and connections between i and i+1(and $2n$-i-3 and $2n$-i-2).

```
Procedure Route_GRN(GN_N, P)
/* Input: A GRN GN_N and a permutation P
        Output: Control setting matrix M[0:2n-2, 0:N-1] */
begin
    for k = 0 to N-1 do    /* Initialize the tag sets I and O */
        I(k) = P(k);
        D(k) = k;
    endfor
    for s = 0 to n-2 do    /* Routing steps from 0 to n-2. */
        for j = 0 to N-1 do
            t = j;
            while (M[s, t] ≠ NIL) do    /* Form a complete
                                            loop */
                M[s, t] = 0;
                u = peer(I(t));
                M[2n-s-2, u] = 0;
                M[2n-s-2, buddy(u)] = 1;
                v = peer(D(buddy(u)));
                M[s, v] = 1;
                t = buddy(v);
            endwhile
        endfor
        Propagate_Tags(i);
    endfor
    for k = 0 to N/2-1 do                /* Step n-1 */
        if (I(2k) = D(2k)) then
            M[n-1, 2k] = 0;
            M[n-1, 2k+1] = 0;
        else
            M[n-1, 2k] = 1;
            M[n-1, 2k+1] = 1;
    endfor
end                    /* End of Routing */
```

Fig. 9. Routing algorithm for a GRN GN$_N$.

For example, the network labeling result for the GRN of Fig. 5(b) is shown in Fig. 8. For a given permutation P such that

$$P = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \\ 12 & 10 & 5 & 4 & 9 & 6 & 15 & 11 & 2 & 8 & 0 & 14 & 1 & 7 & 3 & 13 \end{pmatrix},$$

Fig. 10 shows the routing result of P in the GRN of Fig. 5(b).

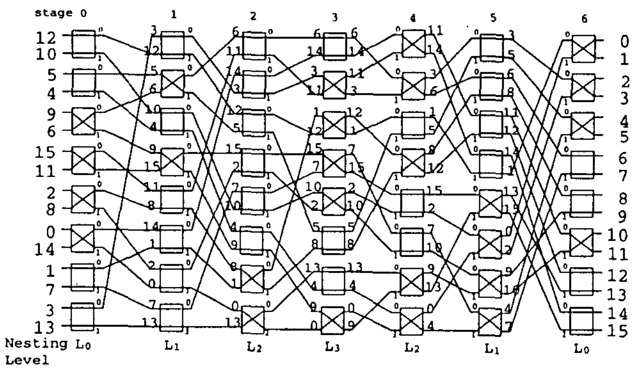


Fig. 10. Routing result of the GRN of Fig. 5(b).

The computation time complexity of the routing algorithm of Fig. 9 is O($N$log$N$). If each switch of the GRN keeps the labels of it and its two ports, the routing algorithm can be parallelized. At step i where $0 \le i \le n$-1, there are 2^i disjoint sub-networks of nesting level L_i such as $GN^0_{2^{n-i}}$, $GN^1_{2^{n-i}}$, $\cdots$, $GN^{2^i-1}_{2^{n-i}}$, which can be routed in parallel.

Ⅳ. Conclusion

The rearrangeability for 2log$N$(or 2log$N$-1) stage MINs were studied by many authors and some of them produced incorrect results [10]. The rearrangeability problem of 2log$N$(or 2log$N$-1) stage MINs needs more theoretical research.

This paper has suggested a class of rearrangeable networks called the generalized rearrangeable networks(GRNs). GRNs are obtained from the Benes network by rearranging the connections between stages and the switches within stages. Networks in this class have the recursive decomposition structure and are controllable by the outside-in decomposition of permutations like the Benes network. The GRNs consist of all of the rearrangeable networks with recursive decomposition structure, and include the rearrangeable networks constructed from the nonequivalent MINs [2]. This paper has also suggested a necessary condition for a network to be a GRN and a network labeling scheme to check whether a network satisfies the condition. The GRNs can be routed by modifying slightly the looping algorithm of the Benes network. The computation time complexity of the routing algorithm is O($N$log$N$). The computation time complexity could be improved by parallelizing the routing algorithm.

References

[1] D. P. Agrawal, "Graph Theoretical Analysis and Design of Muitistage Interconnection Networks," IEEE Trans. on Computers, pp. 637-648, Jul. 1983.

[2] D. P. Agrawal, S. C. Kim, and N. K. Swain, "Analysis and Design of Nonequivalent Muitistage Interconnection Networks," IEEE Trans. on Computers, pp. 232-237, Feb. 1988.

[3] S. Andresen, "The Looping Algorithm Extended to Base 2^t Rearrangeable Switching Networks," IEEE Trans. on Communications, pp. 1057-1063, Oct. 1977.

[4] V. E. Benes, "On Rearrangeable Three-Stage Connecting Networks," Bell Syst. Tech. J., vol. XLI, no. 5, pp. 1481-1492, 1962.

[5] M. K. Kim, H. Yoon, and S. R. Maeng, "Bit-Permute Multistage Interconnection Networks," Microprocessing and Microprogramming 41, pp. 449-468, 1995.

[6] K. Y. Lee, "On the Rearrangeability of 2(log$_2N$)-1 Stage Permutation Networks," IEEE Trans. on Computers, pp. 412-425, May 1985.

[7] C. L. Liu, Introduction to Combinatorial Mathematics, New York: McGraw-Hill, 1968.

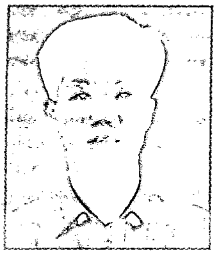
[8] D. C. Opferman and N. T. Tsao-Wu, "On a Class of Rearrangeable Switching Networks-Part I: Control Algorithm," Bell Syst. Tech. J., vol. 50, no. 5, pp. 1579-1600, 1971.

[9] D. S. Parker, "Notes on Shuffle/Exchange-Type Switching Networks," IEEE Trans. on Computers, pp. 213-222, Mar. 1980.

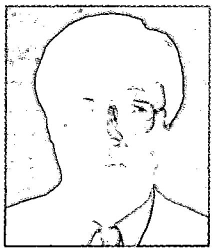
[10] A. Varma and C. S. Raghavendra, Interconnection Networks for Multiprocessors and Multicomputers: Theory and Practice, IEEE Computer Society Press, 1994.

[11] A. Waksman, "A Permutation Network," J. ACM, vol. 15, pp. 159-163, Jan. 1968.

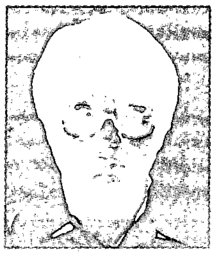
[12] Y. M. Yeh and T. Y. Feng, "On a Class of Rearrangeable Networks," IEEE Trans. on Computers, pp. 1361-1379, Nov. 1992.



Myung-Kyun Kim received the B.S. degree in computer engineering from Seoul National University, Seoul, Korea in 1984, and the M.S. degree in computer science from Korea Advanced Institute of Science and Technology (KAIST) in 1986. He is currently an associate professor in the department of computer engineering, Woosuk University, Cheonbuk, Korea. His research interests include interconnection networks, ATM networks, and intelligent networks.



Hyunsoo Yoon received B.S. degree in electronics engineering from Seoul National University, Seoul, Korea in 1979, M.S. degree in computer science from Korea Advanced Institute of Science and Technology (KAIST) in 1981, and Ph.D. degree in computer and information science from Ohio State University, Columbus, in 1988. From 1978 to 1980, he was with Tongyang Braodcasting Company, Korea. From 1980 to 1894, he was with the computer division of Samsung Electronics Company, Korea, and from 1988 to 1989, he was with AT&T Bell Laboratories as a member of technical staffs. He is currently an associate professor at KAIST. His research interests include parallel computer architectures and communication protocols.



Seung-Ryoul Maeng received B.S. degree in electronics engineering from Seoul National University, Seoul, Korea in 1977, M.S. and Ph.D. degrees in computer science from Korea Advanced Institute of Science and Technology (KAIST) in 1979 and 1984, respectively. Since 1984, he has been a faculty member of the department of computer science of KAIST. From 1988 to 1989, he was with University of Pennsylvania as a visiting scholar. His research interests include parallel computer architectures, vision architectures, and neural networks.