A New Self-Routing Multicast Network

Yuanyuan Yang*
Department of Computer Science
University of Vermont, Burlington, VT 05405

Jianchao Wang
GTE Laboratories
40 Sylvan Road, Waltham, MA 02254

Abstract

In this paper, we propose a design for a new self-routing multicast network which can realize arbitrary multicast communication without any blocking. The network design is based on the binary radix sorting concept and all functional components of the network are recursively constructed reverse banyan networks. The new multicast network we design is compared favorably with the previously proposed multicast networks. It uses $O(n \log^2 n)$ logic gates, and has $O(\log^2 n)$ gate delay and $O(\log^2 n)$ set-up time where the unit of time is a gate delay. Moreover, the good modularity of the network renders a potential to further reduce the network cost by reusing part of the network. For example, the feedback version of our design can reduce the network cost to $O(n \log n)$.

1 Introduction

Multicast communication is one of the most important collective communication operations[1] and is highly demanded in parallel applications as well as in other communication environments. For example, multicast is required to make updates in replicated and distributed databases, multiprocessor systems also require multicast for cache coherence and message passing, and multicast is a critical operation for video/tele-conference calls and video-on-demands services in a telecommunication environment. Clearly, providing multicast support at hardware/interconnection network level is the most efficient way supporting such communication operations[1]. Switching networks that can realize arbitrary multicast communication without any blocking are referred to as *multicast networks*. This type of network has been investigated by several researchers in the literature[2]-[8].

In this paper, we design a new type of multicast network using an approach based on recursive decompositions of multicast networks. This approach was first introduced by Nassimi and Sahni[2] and was later adopted by Lee and Oruç[6] in their design of a multicast network. The $n \times n$ multicast network proposed in [2] uses $O(kn^{1+\frac{1}{k}} \log n)$ 2×2 switches and has $O(k \log n)$ depth and $O(k \log n)$ set-up time for any k, $1 \leq k \leq \log n$. Since the routing algorithm in [2] relies on a cube or a perfect shuffle connected parallel computer consisting of $O(n^{1+\frac{1}{k}})$ processors, as mentioned in [6], the routing process actually takes $O(k \log^2 n)$ gate delays. Lee and Oruç[6] designed a multicast network with a special built-in routing circuit. Their network uses $O(n \log^2 n)$ logic gates, and has $O(\log^2 n)$ gate delay and $O(\log^3 n)$ set-up time where the unit of time is a gate delay.

Another notable feature in our multicast network design is the self-routing scheme, in which the routing circuit is distributed into individual switches. Self-routing is a promising routing scheme which usually renders a network with faster switch setting and less hardware complexity. However, most of self-routing network designs described in the literature are for permutation networks, for example, [9] and [10]. In a recent work, Cheng and Chen[10] designed a new self-routing permutation network constructed by reverse banyan networks.

In this paper, we propose a design for a new self-routing multicast network, which is also based on the reverse banyan networks. We will explore the properties of a reverse banyan network so that it can be used to handle arbitrary multicast connections in a self-routing manner. Different from earlier proposed multicast networks, the new multicast network is conceptually simple and has a good modularity. The network design is based on the binary radix sorting concept and all functional components of the network are recursively constructed reverse banyan networks. Therefore, it renders a potential to greatly reduce the network cost by reusing part of the network. The new multicast network we design uses $O(n \log^2 n)$ logic gates, and has $O(\log^2 n)$ gate delay and $O(\log^2 n)$ set-up time where the unit of time is a gate delay. Moreover, by re-using part of the network, the feedback version of our design can reduce the network cost to $O(n \log n)$.

2 Multicast Networks Based on Binary Radix Sorting

We consider an $n \times n$ interconnection network with n inputs and n outputs where $n = 2^m$ and each input or output address can be expressed as an m-bit binary number $a_0 a_1 \ldots a_{m-1}$. For a multicast connection from network input i ($0 \leq i \leq n-1$) to a subset of network outputs, let I_i denote the subset of the outputs that input i is connected to. I_i is referred to as the *destination set* of the multicast connection, or simply the destination set of input i. Then a *multicast assignment* can be expressed as a vector $(I_0, I_1, \ldots, I_{n-1})$, where, $I_i \cap I_j = \phi$ for $i \neq j$ and $\bigcup_{i=0}^{n-1} I_i \subseteq \{0, 1, \ldots, n-1\}$. Clearly, a *permutation assignment* is a special case of a multicast assignment where every I_i has at most one element.

In this paper, we design a multicast network based on binary radix sorting concept and refer to it as a *binary radix sorting multicast network (BRSMN)*. An $n \times n$ BRSMN is recursively constructed by an $n \times n$ *binary splitting network (BSN)* followed by two $\frac{n}{2} \times \frac{n}{2}$ BRSMNs as shown in Figure 1. For an input i, if all elements in its destination set I_i are in the upper (or lower) half of the network outputs, i.e. the most significant bit of every binary address in I_i is 0 (or 1), there is a single connection from input i via the $n \times n$ BSN to an input of the upper (or lower) $\frac{n}{2} \times \frac{n}{2}$ BRSMN with the same destination set I_i. If some elements in I_i go to the upper half and other elements in I_i go to the lower half, there are two connections from input i via the BSN, each of which goes to the

*Research supported by the U.S. Army Research Office under Grant No. DAAH04-96-1-0234 and by the National Science Foundation under Grant No. OSR-9350540 and MIP-9522532.

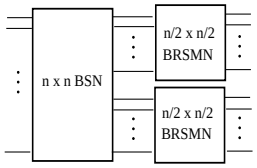

Figure 1: The construction of an $n \times n$ binary radix sorting multicast network (BRSMN).

upper and the lower $\frac{n}{2} \times \frac{n}{2}$ BRSMNs, respectively, and the original destination set I_i is split into two subsets which form the destination sets of the corresponding inputs of the upper and the lower $\frac{n}{2} \times \frac{n}{2}$ BRSMNs, respectively. Then, for an $\frac{n}{2} \times \frac{n}{2}$ BRSMN, we need to check the second most significant bit of the binary addresses in the corresponding destination set, and so on. Finally, for a 2×2 BRSMN (i.e. a 2×2 switch), realizing a multicast or a unicast connection is straightforward. Figure 2 shows the routing for a multicast assignment in an 8×8 BRSMN.

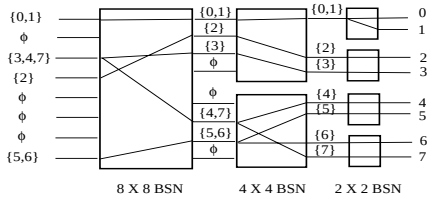

Figure 2: Routing for multicast assignment $(\{0,1\}, \phi, \{3,4,7\}, \{2\}, \phi, \phi, \phi, \{5,6\})$ in an 8×8 BRSMN.

Now the problem of constructing a self-routing multicast network is transformed to the problem of designing a binary splitting network (BSN) described above.

3 The Binary Splitting Network

The binary splitting network splits the multicast connection on each of its inputs into two multicast connections (when necessary) based on the outputs in the destination set of the multicast connection belong to the upper half or the lower half of the network outputs. Since the splitting of an output is determined by the most significant bit of its binary address, the routing in a BSN can be simplified by using a routing tag with four values for each link: 0, 1, α, and ϵ, where, 0 and 1 indicate that all destination addresses of the multicast passing on the link have a 0 and a 1 in the most significant bit, respectively, α means that at least one destination address of the multicast on the link has a 0 in the most significant bit and at least one destination address of the multicast has a 1 in the most significant bit, and ϵ means the link is idle (i.e. carries no packets). The operations on four values in a 2×2 switch are simply an extension to those on only two values 0 and 1. Figure 3 shows all legal operations on four values in a 2×2 switch, where, the operations in Figure 3(c) and (d) are broadcast with values α and ϵ on the inputs changed to 0 and 1 on the outputs.

In an $n \times n$ BSN using the four value routing tags, let n_0, n_1, n_α, and n_ϵ denote the numbers of inputs with value 0, 1, α, ϵ, respectively. It can be verified that they satisfy the constraints:

$$n_0 + n_1 + n_\alpha + n_\epsilon = n \tag{1}$$

$$n_0 + n_\alpha \leq \frac{n}{2} \text{ and } n_1 + n_\alpha \leq \frac{n}{2} \tag{2}$$

$$n_\alpha \leq n_\epsilon. \tag{3}$$

Now, the function of a BSN is to transform the input tags 0's, 1's, α's, and ϵ's to the output side such that all α's are eliminated, all 0's are in the upper half of outputs, all 1's are in the lower half of outputs. Let $\widetilde{n}_0$, $\widetilde{n}_1$, $\widetilde{n}_\epsilon$, and $\widetilde{n}_\alpha$ denote the numbers of outputs with value 0, 1, ϵ, and α, respectively. Since any α paired with one ϵ will be transformed to a pair of 0 and 1, we must have

$$\widetilde{n}_0 = n_0 + n_\alpha, \widetilde{n}_1 = n_1 + n_\alpha, \widetilde{n}_\epsilon = n_\epsilon - n_\alpha, \widetilde{n}_\alpha = 0 \tag{4}$$

$$0 \leq \widetilde{n}_0, \widetilde{n}_1 \leq \frac{n}{2}, \widetilde{n}_0 + \widetilde{n}_1 + \widetilde{n}_\epsilon = n. \tag{5}$$

An $n \times n$ BSN can be constructed by cascading two $n \times n$ reverse banyan networks (RBNs) as shown in Figure 4. The first RBN scatters all α's to 0's and 1's and is referred to as a *scatter network*. The transformation from the inputs to the outputs of the first RBN is $\{0,1,\alpha,\epsilon\}^* \Rightarrow \{0,1,\epsilon\}^*$. The second RBN transfers all 0's to the upper half of the network outputs, and all 1's to the lower half of the network outputs, while each of ϵ's may go to either the upper half or the lower half. The second RBN is referred to as a *quasi-sorting network*. In Figure 5, we show how an input pattern in a BSN is scattered in the first sub-network and then quasi-sorted in the second sub-network.

Next, we will move onto discussing our basic component network, reverse banyan network, and see how it can perform the functions of a scatter network and a quasi-sorting network.

4 The Reverse Banyan Network

An $n \times n$ reverse banyan network (RBN) is recursively constructed by two $\frac{n}{2} \times \frac{n}{2}$ RBNs followed by an $n \times n$ merging network. An $n \times n$ merging network consists of one stage of $\frac{n}{2}$ 2×2 switches, such that both the input and the output links of the stage are connected according to the perfect shuffle interconnection function. The network construction is shown in Figure 6.

We now discuss a useful property of a reverse banyan network. Suppose only two values, say, β and γ, are to be passed in a reverse banyan network. We define an n-bit *circular compact sequence* of β's and γ's as follows:

$$C^n_{s,l;\beta,\gamma} = \begin{cases} \beta^{[s]}\gamma^{[l]}\beta^{[n-s-l]} & \text{if } s+l \leq n \\ \gamma^{[l-n+s]}\beta^{[n-l]}\gamma^{[n-s]} & \text{if } s+l > n \end{cases} \tag{6}$$

where $0 \leq s < n$ and $0 \leq l \leq n$. The real meaning of $C^n_{s,l;\beta,\gamma}$ is that, in an n-bit sequence all l γ-bits are compacted together followed by also compacted $n-l$ β-bits in a circular way (modulo n), and s is the starting position for the γ-bit sequence. Cheng and Chen[10] considered the circular compact sequence of 0's and 1's in their permutation network, and found an interesting property. We state it below in a more general form.

Theorem 1 *For any β-γ values on the inputs of an RBN, a circular compact sequence with any starting position can be achieved at the outputs of the RBN under a proper setting for switches in the network.*

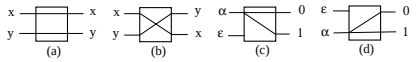

Figure 3: Legal operations on the four values in a 2×2 switch, where $x, y \in \{0,1,\alpha,\epsilon\}$.

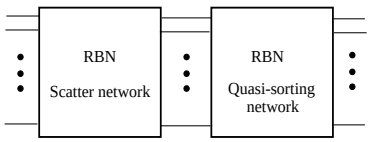

Figure 4: The construction of a binary splitting network.

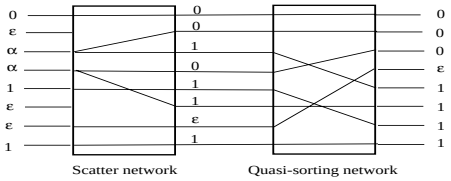

Figure 5: Data scattered in the first sub-network and then quasi-sorted in the second sub-network of a BSN.

In the following, we provide a different, much easier to understand proof for Theorem 1. The technique used in the proof and some observations will be applied to the design of our multicast network in later sections.

Suppose the inputs of the $n \times n$ RBN consist of l γ's and $n-l$ β's, among which the upper half $\frac{n}{2}$ inputs contain l_0 γ's and the lower half $\frac{n}{2}$ inputs contain l_1 γ's, where $l_0 + l_1 = l$. Assume Theorem 1 holds for an $\frac{n}{2} \times \frac{n}{2}$ RBN. We will prove that it also holds for an $n \times n$ RBN by giving a positive answer to the following question.

Question 1 *Given n, s, l, l_0, and l_1 satisfying that n is an even number, $0 \le s < n$, $0 \le l \le n$, $0 \le l_0, l_1 \le \frac{n}{2}$, and $l = l_0 + l_1$, do there exist integers s_0 and s_1, $0 \le s_0, s_1 < \frac{n}{2}$, such that $C^{n/2}_{s_0,l_0;\beta,\gamma}$ and $C^{n/2}_{s_1,l_1;\beta,\gamma}$ can be merged to $C^{n}_{s,l;\beta,\gamma}$ through an $n \times n$ merging network (as defined in this section) under a proper switch setting (in one-to-one mapping)?*

Before we answer this question, we first review some observations on the shuffle/exchange interconnection functions. For a binary number $a = a_{m-1}a_{m-2}\ldots a_0$, we have *shuffle*$(a) = a_{m-2}\ldots a_0 a_{m-1}$ and *exchange*$(a) = a_{m-1}a_{m-2}\ldots \overline{a}_0$. Consider an $n \times n$ merging network whose inputs (outputs) are linked to switches according to the perfect shuffle function. Let a be a binary address of an input of a switch (with address $\lfloor \frac{a}{2} \rfloor$) in the network. Then *exchange*(a) denoted as $\overline{a}$ is the other input of the switch. The inputs of the merging network with addresses *shuffle*(a) and *shuffle*$(\overline{a})$ are linked to inputs a and $\overline{a}$ of the switch, respectively, and the outputs a and $\overline{a}$ of the switch are linked to the outputs of the

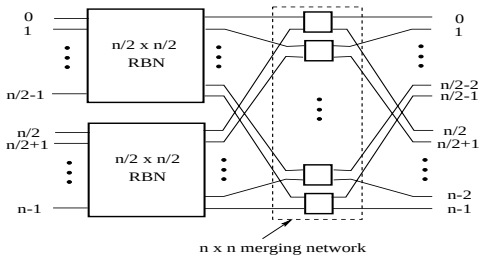

Figure 6: The recursive definition of an $n \times n$ RBN. The dashed box is an $n \times n$ merging network.

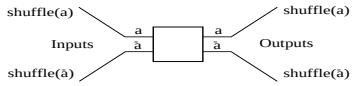

Figure 7: The connections through a switch in a merging network, where $\overline{a} =$ exchange(a).

merging network with addresses *shuffle*(a) and *shuffle*$(\overline{a})$, respectively. (See Figure 7.) We can see that connections are only possible between the inputs and the outputs of a merging network with addresses *shuffle*(a) and *shuffle*$(\overline{a})$. Since only one-to-one mapping is considered, the switch has only two settings: parallel and crossing. Note that $|$*shuffle*$(a) -$ *shuffle*$(\overline{a})| = \frac{n}{2}$, that is, two inputs with $\frac{n}{2}$ apart in their addresses can be connected to outputs with the same addresses in a parallel or crossing way. Let r_i denote the setting for switch i ($r_i = 0$: set to parallel; $r_i = 1$: set to crossing). We can also use a circular compact sequence to represent switch setting. To avoid confusion in notations, we use $W^{n/2}_{s,l;0,1}$ to specify the compact switch setting of $\frac{n}{2}$ switches in an $n \times n$ merging network, where l consecutive switches have setting 1 with starting position s, and the rest of switches have setting 0 in a circular way.

The following lemma gives a positive answer to Question 1.

Lemma 1 *Given n, s, l, l_0, and l_1 satisfying that n is an even number, $0 \le s < n$, $0 \le l \le n$, $0 \le l_0, l_1 \le \frac{n}{2}$, and $l = l_0 + l_1$. Let $s_0 = s \bmod \frac{n}{2}$, $s_1 = (s + l_0) \bmod \frac{n}{2}$ and the switch setting of an $n \times n$ merging network be*

$$r_i = \begin{cases} b & 0 \le i < s_1 \\ \overline{b} & s_1 \le i \le \frac{n}{2} \end{cases}$$

where $b = \left[(s + l_0) \textbf{ div } \frac{n}{2}\right] \bmod 2$, and $\overline{b} = (1 - b) \bmod 2$. Then $C^{n/2}_{s_0,l_0;\beta,\gamma}$ and $C^{n/2}_{s_1,l_1;\beta,\gamma}$ can be merged to $C^{n}_{s,l;\beta,\gamma}$ through the $n \times n$ merging network under the above switch setting. Using the notations of the circular compact sequence, this merging operation in the $n \times n$ merging network can be expressed as

$$C^{n/2}_{s_0,l_0;\beta,\gamma} C^{n/2}_{s_1,l_1;\beta,\gamma} \xrightarrow{W^{n/2}_{0,s_1;\overline{b},b}} C^{n}_{s,l;\beta,\gamma}$$

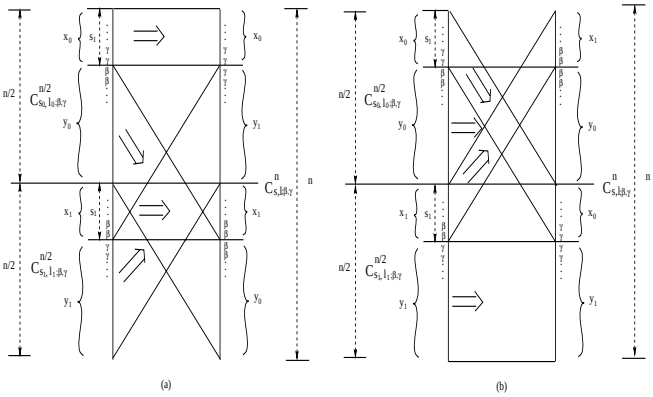

Figure 8: Illustration of the proof of Lemma 1.

Proof of Lemma 1. We now prove that the given two $\frac{n}{2}$-bit circular compact sequences $C^{n/2}_{s_0,l_0;\beta,\gamma}$ and $C^{n/2}_{s_1,l_1;\beta,\gamma}$ can be merged to $C^{n}_{s,l;\beta,\gamma}$ through an $n \times n$ merging network by the given switch setting. In Figure 8(a) (or 8(b)), on the left side both $C^{n/2}_{s_0,l_0;\beta,\gamma}$ and $C^{n/2}_{s_1,l_1;\beta,\gamma}$ are expressed as vertical segments of length $\frac{n}{2}$, and on

the right side $C^n_{s,l;\beta,\gamma}$ is expressed as a vertical segment of length n. Note that the first s_1 consecutive switches all are set to setting b, and the rest of $\frac{n}{2} - s_1$ consecutive switches are set to the opposite setting $\overline{b}$. Thus the segment for $C^{n/2}_{s_0,l_0;\beta,\gamma}$ can be divided into two sub-segments x_0 (of length s_1) and y_0 (of length $\frac{n}{2} - s_1$). Similarly, the segment for $C^{n/2}_{s_1,l_1;\beta,\gamma}$ can be divided into two sub-segments x_1 (of length s_1) and y_1 (of length $\frac{n}{2} - s_1$). Let symbol $\circ$ denote a concatenation operation. Note that segment $x_1 \circ y_1$ represents $C^{n/2}_{s_1,l_1;\beta,\gamma}$ and x_1 is of length s_1 which equals the starting position of the γ's sequence in $C^{n/2}_{s_1,l_1;\beta,\gamma}$. We must have that sequence x_1 ends with β and sequence y_1 starts with γ. Also, since $s_1 = (s + l_0) \bmod \frac{n}{2} = (s_0 + l_0) \bmod \frac{n}{2}$ and x_0 is of length s_1, in segment $x_0 \circ y_0$ we must have that sequence x_0 ends with γ and sequence y_0 starts with β. We consider two cases of b values.

Case 1: $b = 0$. As shown in Figure 8(a), all inputs in both sub-segments x_0 and x_1 on the left are routed to the outputs on the right in a parallel way, and all inputs in both sub-segments y_0 and y_1 on the left are routed to the outputs on the right in a crossing way. Therefore, the segment on the right is $x_0 \circ y_1 \circ x_1 \circ y_0$. We now prove that the sequence represented by the segment on the right is indeed the circular compact sequence $C^n_{s,l;\beta,\gamma}$. In the segment $x_0 \circ y_1 \circ x_1 \circ y_0$, x_0 ending with γ and y_1 starting with γ make the γ's in $C^{n/2}_{s_0,l_0;\beta,\gamma}$ and $C^{n/2}_{s_1,l_1;\beta,\gamma}$ consecutive at the joint of x_0 and y_1. Also, x_1 ending with β and y_0 starting with β make the β's in $C^{n/2}_{s_1,l_1;\beta,\gamma}$ and $C^{n/2}_{s_0,l_0;\beta,\gamma}$ consecutive at the joint of x_1 and y_0. Moreover, the consecutiveness of β or γ is preserved at the joint of y_1 and x_1, since $x_1 \circ y_1$ is a circular compact sequence. A similar argument applies to the joint of y_0 and x_0. Thus, the segment $x_0 \circ y_1 \circ x_1 \circ y_0$ is a circular compact sequence since l_0 γ's from $C^{n/2}_{s_0,l_0;\beta,\gamma}$ and l_1 γ's from $C^{n/2}_{s_1,l_1;\beta,\gamma}$ are concatenated in the segment. Finally we can check that the starting position for γ's sequence in $x_0 \circ y_1 \circ x_1 \circ y_0$ is s. Thus the merged sequence is $C^n_{s,l;\beta,\gamma}$. Note that $b = 0$ implies $\left[(s + l_0) \textbf{ div } \frac{n}{2}\right] \bmod 2 = 0$. It follows that in this case we must have either $s + l_0 < \frac{n}{2}$ or $s + l_0 \geq n$ (of course, $s + l_0 < \frac{3n}{2}$). Also from $s_1 = (s + l_0) \bmod \frac{n}{2}$, we have $s = (s_1 - l_0) \bmod n$. In the sequence $x_0 \circ y_1 \circ x_1 \circ y_0$, the ending point (i.e. bottom) of x_0 is at position s_1. From this point going upward (in a circular way), there are consecutive l_0 γ's, and the last γ is at the starting position for γ's sequence in $x_0 \circ y_1 \circ x_1 \circ y_0$, which is $(s_1 - l_0) \bmod n$ in this case.

Case 2: $b = 1$. The proof is similar (using Figure 8(b)). □

Lemma 1 is actually the inductive step of the proof of Theorem 1, and it is easy to check that for $n = 2$ Theorem 1 holds. That proves Theorem 1.

Note that for a full permutation assignment, only two tag values 0 and 1 are used. Let β be 0, γ be 1, and the initial starting position for an $n \times n$ RBN be $s = \frac{n}{2}$. Clearly the total number of 1's in this case is $l = \frac{n}{2}$. By Theorem 1, the circular compact sequence $C^n_{s,l;0,1} = 0^{\lfloor \frac{n}{2} \rfloor} 1^{\lceil \frac{n}{2} \rceil}$, which represents the function of bit sorting, can be achieved by a proper switch setting.

5 The RBN as a Scatter Network

In the last section, we considered only two values (0 and 1) for a full permutation assignment in an RBN. For a multicast assignment, we must deal with four values 0, 1, α, and ϵ. The following theorem gives the main result of this section, and its proof is given at the end of the section.

Theorem 2 *Given an $n \times n$ RBN, which is the scatter network of an $n \times n$ BSN, with 0, 1, α and ϵ values on the inputs, then under a proper setting for switches in the network, α's can be eliminated at the outputs of the RBN, that is, the outputs of the RBN have only values 0, 1, and ϵ and satisfying (4) and (5).*

It is worth pointing out that although $n_\alpha \leq n_\epsilon$ holds globally for the original $n \times n$ RBN which is the scatter network of an $n \times n$ BSN (see (3)), for the recursively defined sub-network $n' \times n'$ RBN of the $n \times n$ RBN, we may have $n'_\alpha \geq n'_\epsilon$, where n'_α and n'_ϵ are the numbers of inputs (of this $n' \times n'$ RBN) with values α and ϵ respectively. This is because that α's and ϵ's are distributed non-uniformly. To simplify the problem, we can combine 0 and 1 into a single value χ. A link has a value χ if it has a single value 0 or 1. If the two inputs of a 2×2 switch have values α and ϵ respectively, α can be scattered so that the two outputs of the switch have values χ's. Let n_α, and n_ϵ be the numbers of inputs with value α and ϵ, respectively. By exploring the properties of the circular compact sequence, we can obtain the following general results for an $n \times n$ RBN with any numbers of α's and ϵ's on its inputs.

Theorem 3 *For any values on the inputs of an $n \times n$ RBN, if $n_\alpha \leq n_\epsilon$ (or $n_\alpha \geq n_\epsilon$), a circular compact sequence $C^n_{s,n_\epsilon - n_\alpha;\chi,\epsilon}$ (or $C^n_{s,n_\alpha - n_\epsilon;\chi,\alpha}$) with any starting position s ($0 \leq s < n$) can be achieved at the outputs of the RBN under a proper setting for switches in the network.*

In the above theorem, we say that ϵ (or α) is the dominating type among ϵ and α if $n_\alpha \leq n_\epsilon$ (or $n_\alpha \geq n_\epsilon$).

We now further explore the property of the circular compact sequence in order to deal with multicast assignments. In the case of merging two circular compact sequences with the same set of binary values, (e.g. merging two sequences with χ's and α's, $C^{n/2}_{s_0,l_0;\chi,\alpha}$ and $C^{n/2}_{s_1,l_1;\chi,\alpha}$), we can simply use the results in Theorem 1. However, in other cases, we may also need to merge two circular compact sequences with different sets of binary values (e.g. merging a sequence with χ's and α's, $C^{n/2}_{s_0,l_0;\chi,\alpha}$, and a sequence with χ's and ϵ's, $C^{n/2}_{s_1,l_1;\chi,\epsilon}$). In this case, we can add two more switch settings: upper broadcast and lower broadcast, as shown in Figure 3(c) and 3(d). For any 2×2 switch i, in addition to switch settings $r_i = 0$ or 1, let the switch setting $r_i = 2$ if the switch is set to upper broadcast, and $r_i = 3$ if the switch is set to lower broadcast. To merge two compact sequences with different sets of binary values, we need to answer the following question.

Question 2 *Given n, s, l, l_0, and l_1 satisfying that n is an even number, $0 \leq s < n$, $0 \leq l \leq n$, $0 \leq l_1 \leq l_0 \leq \frac{n}{2}$, and $l = l_0 - l_1$, do there exist integers s_0 and s_1 ($0 \leq s_0, s_1 < \frac{n}{2}$) such that $C^{n/2}_{s_0,l_0;\chi,\alpha}$ and $C^{n/2}_{s_1,l_1;\chi,\epsilon}$ can be merged to $C^n_{s,l;\chi,\alpha}$ through an $n \times n$ merging network under a proper switch setting?*

First, we need to extend the concept of the circular compact sequence with binary values to that with trinary values in the context of switch setting. A trinary circular compact sequence of switch setting $W^{n/2}_{s,l_1,l_2;\beta_1,\beta_2,\beta_3}$ means l_1 consecutive β_2's followed by l_2 consecutive β_3's and then followed by $\frac{n}{2} - l_1 - l_2$ β_1's in a circular way, with s being the starting position of the β_2's sequence. The following lemma gives a positive answer to Question 2.

Lemma 2 *Given n, s, l, l_0, and l_1 satisfying that n is an even number, $0 \leq s < n$, $0 \leq l \leq n$, $0 \leq l_1 \leq l_0 \leq \frac{n}{2}$, and $l = l_0 - l_1$. Let $s_0 = s \bmod \frac{n}{2}$, $s_1 = (s + l) \bmod \frac{n}{2}$ and the switch setting be*

(1) $W^{n/2}_{s_1,l_1;0,2}$*, if* $s+l<\frac{n}{2}$*;*
(2) $W^{n/2}_{s_1,l_1,\frac{n}{2}-s_1-l_1;1,2,0}$*, if* $s<\frac{n}{2}$ *and* $s+l\geq\frac{n}{2}$*;*
(3) $W^{n/2}_{s_1,l_1;1,2}$*, if* $s\geq\frac{n}{2}$ *and* $s+l<n$*;*
(4) $W^{n/2}_{s_1,l_1,\frac{n}{2}-s_1-l_1;0,2,1}$*, if* $s\geq\frac{n}{2}$ *and* $s+l\geq n$*.*

Then $C^{n/2}_{s_0,l_0;\chi,\alpha}$ *and* $C^{n/2}_{s_1,l_1;\chi,\epsilon}$ *can be merged to* $C^{n}_{s,l;\chi,\alpha}$ *through an* $n\times n$ *merging network under the above switch setting.*

Proof of Lemma 2. It should be pointed out that since $0\leq l\leq l_0$, the four cases in Lemma 2 (i.e. (1) $s+l<\frac{n}{2}$, (2) $s<\frac{n}{2}$ and $s+l\geq\frac{n}{2}$, (3) $s\geq\frac{n}{2}$ and $s+l<n$, and (4) $s\geq\frac{n}{2}$ and $s+l\geq n$) cover all possible intervals. Applying a similar approach to the proof of Lemma 1, we represent the input and output sequences as vertical segments as shown in Figure 9(a)-(d), with one sub-figure for each of the four cases.

Note that all α's on the inputs are in the upper half and all ϵ's on the inputs are in the lower half. Let's first look at the two sub-segments y_0 and y_1 on the left in Figure 9(a). In the lower half of the inputs (i.e. $C^{n/2}_{s_1,l_1;\chi,\epsilon}$), the sub-segment y_1 which starts from position s_1 and is of length l_1 (in a circular way, modulo $\frac{n}{2}$) consists of consecutive ϵ's. Also, since
$s_1=(s+l) \bmod \frac{n}{2}=(s_0+l) \bmod \frac{n}{2}=(s_0+l_0-l_1) \bmod \frac{n}{2}$,
in the upper half of the inputs (i.e. $C^{n/2}_{s_0,l_0;\chi,\alpha}$), the sub-segment y_0 which starts from position s_1 and is of length l_1 consists of consecutive α's. Thus, these two sub-segments from the upper and lower halves of the inputs are at the same position within their segment and have a distance of $\frac{n}{2}$. A similar observation can be drawn for Figure 9(b)-(d). In addition, in the switch settings (1)-(4) in Lemma 2, only those l_1 consecutive switches (in a circular way) starting from the s_1^{th} switch have an upper broadcast setting. That is, all corresponding α's and ϵ's in these two sub-segments y_0 and y_1 are neutralized (or eliminated) and become χ's in the corresponding positions on the outputs of the RBN.

The rest of the consecutive α's on the inputs form a sub-segment named x_0 as shown in Figure 9(a)-(d). x_0 starts from position s_0, ends at position s_1-1 (in a circular way, modulo $\frac{n}{2}$) in the upper half, and is of length $l=l_0-l_1$. Next, we will show how the α's in x_0 are mapped to the pre-determined positions in the outputs of the RBN case by case.

Case 1: $s+l<\frac{n}{2}$. See Figure 9(a). In this case, since $s+l<\frac{n}{2}$, all l α's on the outputs will be in the upper half. Note that in switch setting $W^{n/2}_{s_1,l_1;0,2}$, all the switches except those with an upper broadcast setting have a parallel setting. Since the sub-segment x_0 (consisting of all α's) in the upper half of the inputs is mapped to the outputs in a parallel way, the l consecutive outputs starting from $s=s_0$ have α's, and the rest of the outputs have χ's. Hence, the entire sequence of the outputs is $C^{n}_{s,l;\chi,\alpha}$.

Case 2: $s<\frac{n}{2}$ and $s+l\geq\frac{n}{2}$. See Figure 9(b). In this case, the segment of the consecutive outputs with α's will go across the middle point of the entire segment of all n outputs. However, all α's on the left are in the upper half of the inputs. This indicates that we need to map some α's on the inputs to the outputs in a parallel way and map some α's in a crossing way. The switch setting $W^{n/2}_{s_1,l_1,\frac{n}{2}-s_1-l_1;1,2,0}$ can accomplish this. We can see that the sub-segment x_0 (consisting of all α's) in the upper half of the inputs can be divided into two parts. Since $s_0=s$ and $s_1=s+l-\frac{n}{2}$, the sub-segment of the inputs with α's starting at $s_0=s$ and ending at $\frac{n}{2}-1$ is mapped to the outputs in a parallel way, and the sub-segment of the inputs with α's starting at 0 and ending at s_1-1 is mapped to the outputs in a crossing way, that is, mapped to a sub-segment of the outputs starting at $\frac{n}{2}$ and ending at $\frac{n}{2}+s_1-1$. In other words, we obtain that $l=s_1+\frac{n}{2}-s$ consecutive outputs have α's starting at $s=s_0$, and the rest of the outputs have χ's, which is $C^{n}_{s,l;\chi,\alpha}$.

Case 3: $s\geq\frac{n}{2}$ and $s+l<n$. The proof is similar to Case 1. (See Figure 9(c).)

Case 4: $s\geq\frac{n}{2}$ and $s+l\geq n$. The proof is similar to Case 2. (See Figure 9(d).) □

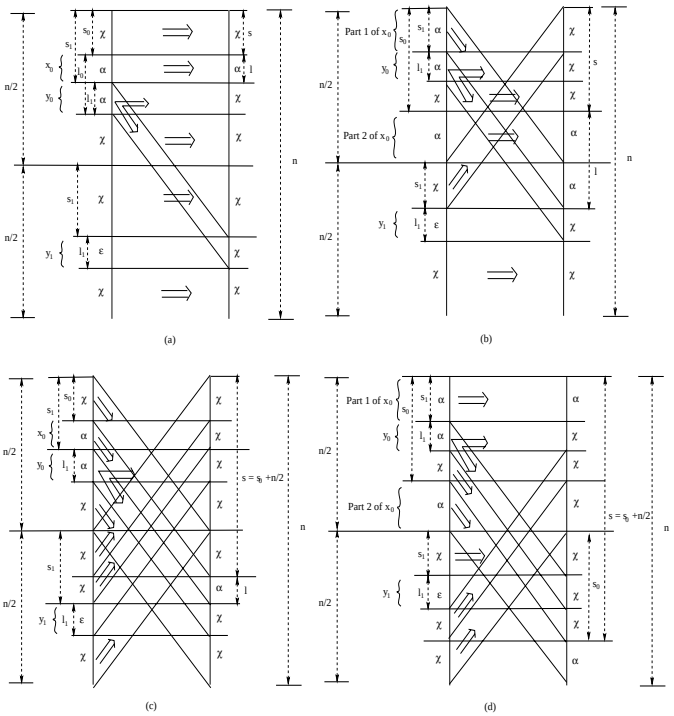

Figure 9: The illustration of the proof of Lemma 2.

Question 2 has other three symmetric variants, which can be obtained by changing the condition in Lemma 2 to $l_1\geq l_0$ and $l=l_1-l_0$, and/or swapping α for ϵ. The corresponding solutions to these three variants are also symmetric. Lemmas 3 - 5 give such solutions.

Lemma 3 *Given* n*,* s*,* l*,* l_0*, and* l_1 *satisfying that* n *is an even number,* $0\leq s<n$*,* $0\leq l\leq n$*,* $0\leq l_0\leq l_1\leq\frac{n}{2}$*, and* $l=l_1-l_0$*. Let* $s_0=(s+l) \bmod \frac{n}{2}$*,* $s_1=s \bmod \frac{n}{2}$ *and the switch setting be*
(1) $W^{n/2}_{s_0,l_0;1,2}$*, if* $s+l<\frac{n}{2}$*;*
(2) $W^{n/2}_{s_0,l_0,\frac{n}{2}-s_0-l_0;0,2,1}$*, if* $s<\frac{n}{2}$ *and* $s+l\geq\frac{n}{2}$*;*
(3) $W^{n/2}_{s_0,l_0;0,2}$*, if* $s\geq\frac{n}{2}$ *and* $s+l<n$*;*
(4) $W^{n/2}_{s_0,l_0,\frac{n}{2}-s_0-l_0;1,2,0}$*, if* $s\geq\frac{n}{2}$ *and* $s+l\geq n$*,*

then $C^{n/2}_{s_0,l_0;\chi,\alpha}$ *and* $C^{n/2}_{s_1,l_1;\chi,\epsilon}$ *can be merged to* $C^{n}_{s,l;\chi,\epsilon}$ *through an* $n\times n$ *merging network under the above switch setting.*

Lemma 4 *Given* n*,* s*,* l*,* l_0*, and* l_1 *satisfying that* n *is an even number,* $0\leq s<n$*,* $0\leq l\leq n$*,* $0\leq l_1\leq l_0\leq\frac{n}{2}$*, and* $l=l_0-l_1$*. Let* $s_0=s \bmod \frac{n}{2}$*,* $s_1=(s+l) \bmod \frac{n}{2}$ *and the switch setting be*
(1) $W^{n/2}_{s_1,l_1;0,3}$*, if* $s+l<\frac{n}{2}$*;*

(2) $W^{n/2}_{s_1,l_1,\frac{n}{2}-s_1-l_1;1,3,0}$, *if* $s < \frac{n}{2}$ *and* $s+l \geq \frac{n}{2}$;
(3) $W^{n/2}_{s_1,l_1;1,3}$, *if* $s \geq \frac{n}{2}$ *and* $s+l < n$;
(4) $W^{n/2}_{s_1,l_1,\frac{n}{2}-s_1-l_1;0,3,1}$, *if* $s \geq \frac{n}{2}$ *and* $s+l \geq n$,
then $C^{n/2}_{s_0,l_0;\chi,\epsilon}$ *and* $C^{n/2}_{s_1,l_1;\chi,\alpha}$ *can be merged to* $C^{n}_{s,l;\chi,\epsilon}$ *through an* $n \times n$ *merging network under the above switch setting.*

Lemma 5 *Given* n, s, l, l_0, *and* l_1 *satisfying that* n *is an even number,* $0 \leq s < n$, $0 \leq l \leq n$, $0 \leq l_0 \leq l_1 \leq \frac{n}{2}$, *and* $l = l_1 - l_0$. *Let* $s_0 = (s+l) \bmod \frac{n}{2}$, $s_1 = s \bmod \frac{n}{2}$ *and the switch setting be*
(1) $W^{n/2}_{s_0,l_0;1,3}$, *if* $s+l < \frac{n}{2}$;
(2) $W^{n/2}_{s_0,l_0,\frac{n}{2}-s_0-l_0;0,3,1}$, *if* $s < \frac{n}{2}$ *and* $s+l \geq \frac{n}{2}$;
(3) $W^{n/2}_{s_0,l_0;0,3}$, *if* $s \geq \frac{n}{2}$ *and* $s+l < n$;
(4) $W^{n/2}_{s_0,l_0,\frac{n}{2}-s_0-l_0;1,3,0}$, *if* $s \geq \frac{n}{2}$ *and* $s+l \geq n$,
then $C^{n/2}_{s_0,l_0;\chi,\epsilon}$ *and* $C^{n/2}_{s_1,l_1;\chi,\alpha}$ *can be merged to* $C^{n}_{s,l;\chi,\alpha}$ *through an* $n \times n$ *merging network under the above switch setting.*

Now we are in the position to prove Theorem 3.

Proof of Theorem 3. By induction on the network size n.

For $n = 2$ (base case), the network is a 2×2 switch. It is easy to check that Theorem 3 holds in this case (see [11]).

Now assume Theorem 3 holds for an $\frac{n}{2} \times \frac{n}{2}$ RBN (inductive hypothesis) and consider an $n \times n$ RBN. In the recursive definition of an $n \times n$ RBN, let n'_α and n'_ϵ denote the numbers of α's and ϵ's respectively in the upper $\frac{n}{2} \times \frac{n}{2}$ RBN, and n''_α and n''_ϵ denote the numbers of α's and ϵ's in the lower $\frac{n}{2} \times \frac{n}{2}$ RBN respectively. Clearly, we have $n'_\alpha + n''_\alpha = n_\alpha$ and $n'_\epsilon + n''_\epsilon = n_\epsilon$. We first assume $n_\alpha \leq n_\epsilon$ and consider the following cases.

Case 1: $n'_\alpha \leq n'_\epsilon$ and $n''_\alpha \leq n''_\epsilon$. By the inductive hypothesis, Theorem 3 holds for both upper and lower $\frac{n}{2} \times \frac{n}{2}$ RBNs. That is, for any given integers s_0 and s_1 ($0 \leq s_0, s_1 < \frac{n}{2}$), $C^{n/2}_{s_0,n'_\epsilon - n'_\alpha;\chi,\epsilon}$ and $C^{n/2}_{s_1,n''_\epsilon - n''_\alpha;\chi,\epsilon}$ can be achieved at the outputs of the upper and lower $\frac{n}{2} \times \frac{n}{2}$ RBNs, respectively. Now, let $l_0 = n'_\epsilon - n'_\alpha$, $l_1 = n''_\epsilon - n''_\alpha$, and $l = n_\epsilon - n_\alpha$, which implies $l = l_0 + l_1$. Then by Lemma 1, given any integer s ($0 \leq s < n$), there exist integers s_0 and s_1 ($0 \leq s_0, s_1 < \frac{n}{2}$) such that $C^{n/2}_{s_0,l_0;\chi,\epsilon}$ and $C^{n/2}_{s_1,l_1;\chi,\epsilon}$ can be merged to $C^{n}_{s,l;\chi,\epsilon}$ through the $n \times n$ merging network under a proper switch setting. Hence, Theorem 3 holds.

Case 2: $n'_\alpha \leq n'_\epsilon$ and $n''_\alpha \geq n''_\epsilon$. By the inductive hypothesis, for any given integers s_0 and s_1 ($0 \leq s_0, s_1 < \frac{n}{2}$), $C^{n/2}_{s_0,n'_\epsilon - n'_\alpha;\chi,\epsilon}$ and $C^{n/2}_{s_1,n''_\alpha - n''_\epsilon;\chi,\alpha}$ can be achieved at the outputs of the upper and lower $\frac{n}{2} \times \frac{n}{2}$ RBNs, respectively. Let $l_0 = n'_\epsilon - n'_\alpha$, $l_1 = n''_\alpha - n''_\epsilon$, and $l = n_\epsilon - n_\alpha$. Then we have $l = l_0 - l_1$ and $l_0 \geq l_1$. By Lemma 4, given any integer s ($0 \leq s < n$), there exist integers s_0 and s_1 ($0 \leq s_0, s_1 < \frac{n}{2}$) such that $C^{n/2}_{s_0,l_0;\chi,\epsilon}$ and $C^{n/2}_{s_1,l_1;\chi,\alpha}$ can be merged to $C^{n}_{s,l;\chi,\epsilon}$ through the $n \times n$ merging network under a proper switch setting. Thus, the result is also true in this case.

Case 3: $n'_\alpha \geq n'_\epsilon$ and $n''_\alpha \leq n''_\epsilon$. Similar to Case 2, by the inductive hypothesis and Lemma 3, we can see Theorem 3 holds.

Thus, we have proved Theorem 3 for $n_\alpha \leq n_\epsilon$. Symmetrically, we can show that the theorem holds for $n_\alpha \geq n_\epsilon$. □

In the proof of Theorem 3, we can see that the number of the tag values of dominating type in the outputs of an RBN is the sum of those in its two sub RBNs, if the two sub RBNs have the same dominating type. In this case, Lemma 1 is applied, and we refer to it as ϵ/α-addition. On the other hand, the number of the tag values of dominating type is the difference between those of its two sub RBNs, if the two sub RBNs have different dominating types. In this case, Lemma 2-5 are applied, and we refer to it as ϵ/α-elimination.

Finally, we are in the position to prove Theorem 2.

Proof of Theorem 2. Since $n_\alpha \leq n_\epsilon$ holds for the original $n \times n$ RBN which is the scatter network of an $n \times n$ BSN (see (3)), by Theorem 3 we can always eliminate α's at the outputs of the RBN under a proper switch setting. On the other hand, from the proof of Theorem 3, we know that a tag value 0 or 1, once presented on a link at some stage of the RBN, will be passed in a unicast way to some output without any value change. We also know that value changes among 0's, 1's, α's, and ϵ's occur only in some 2×2 switches. In this case, two inputs of the switch have α and ϵ respectively, the switch setting is upper or lower broadcast (this is always true in our design), and two outputs of the switch have 0 and 1 respectively. Consequently, a pair of α and ϵ are transformed to a pair of 0 and 1. In total, n_α such pairs are transformed. Hence, $\widetilde{n}_0$, $\widetilde{n}_1$, $\widetilde{n}_\epsilon$, and $\widetilde{n}_\alpha$, which are the numbers of outputs with value 0, 1, ϵ, and α, respectively, satisfy the following: $\widetilde{n}_0 = n_0 + n_\alpha$, $\widetilde{n}_1 = n_1 + n_\alpha$, $\widetilde{n}_\epsilon = n_\epsilon - n_\alpha$, and $\widetilde{n}_\alpha = 0$. Also by using (1) and (2), we have that $0 \leq \widetilde{n}_0, \widetilde{n}_1 \leq \frac{n}{2}$ and $\widetilde{n}_0 + \widetilde{n}_1 + \widetilde{n}_\epsilon = n$. □

6 The RBN as a Quasi-Sorting Network

Theorem 1 can be used to perform bit sorting in an RBN, that is, under a proper switch setting, all 0's and 1's on the inputs of the RBN can be routed to its outputs in an ascending order. However, this works only for full permutation assignments, in which each input of the RBN has a tag value either 0 or 1, and cannot be directly applied to partial permutation or multicast assignments.

In the following, we discuss how an RBN can be used as a quasi-sorting network for partial permutation or multicast assignments. As can be seen in the last section, the outputs of the $n \times n$ RBN as a scatter network have tag values 0, 1, and ϵ only, which are passed to the inputs of the $n \times n$ RBN as a quasi-sorting network, and the numbers of such 0's or 1's are no more than $\frac{n}{2}$. The function of an $n \times n$ RBN as a quasi-sorting network is to route all 0's and 1's on the inputs of the RBN to the upper and lower halves of its outputs respectively, and to route ϵ's to the remaining positions at the outputs. We can let some of ϵ's be dummy 0's (denoted as ϵ_0's) and the rest of ϵ's be dummy 1's (denoted as ϵ_1's), such that the number of all 0's (including all the real 0's and the dummy 0's) and the number of all 1's (including all the real 1's and the dummy 1's) are equal to $\frac{n}{2}$. Then by applying the bit sorting results in Theorem 1 we can achieve the quasi-sorting.

7 The Self-Routing Algorithms for RBNs

Lemmas 1-5 actually provide a way to perform switch setting for all the switches in an RBN as a bit sorting network and as a scatter network, while switch setting for an RBN as a quasi-sorting network can be transformed to that for an RBN as a bit sorting network after all the ϵ's on the inputs are properly divided into ϵ_0's and ϵ_1's. Based on the recursive construction of an RBN, we can formulate the structure of an RBN into a complete binary tree shown in Figure 10(a). The root node of the tree represents the original RBN as a bit sorting network, a scatter network, or a quasi-sorting network; the two child nodes of the root represent the two recursively defined sub RBN networks of the original RBN; and so on; and finally the leaves of the tree represent the inputs of the original RBN. The distributed routing algorithms are performed by each node of the binary tree. The algorithms start from leaves and

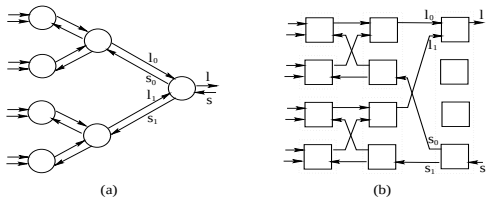

Figure 10: (a) The forward and backward phases in a binary tree. (b) The forward and backward trees embedded in a reverse banyan network.

perform forward computations all the way to the root, and then start from root and perform backward computations all the way to the leaves. The detailed description for the distributed algorithms can be found in [11].

8 Circuit Design Issues and the Complexity

Based on the distributed algorithms, we can design a self-routing circuit for our RBN-based multicast network in a similar approach to Cheng and Chen's[10] self-routing circuit design for their RBN-based permutation network. In this section, we only briefly discuss some circuit design issues.

First of all, as can be seen, five different tag values 0, 1, α, and ϵ (including ϵ_0 and ϵ_1) are used in the RBN as a scatter network and as a quasi-sorting networks. We need to use 3 bits to represent a tag value. The encoding scheme for these tag values is shown in [11]. Secondly, the binary tree structure in all distributed algorithms can be embedded into an RBN. For a balanced hardware distribution, we separate the tree to a forward tree and a backward tree as shown in Figure 10(b), where the first and the last switches of the last stage of a sub RBN network can serve as the nodes in the forward tree and the backward tree, respectively, and the switches in between can use the results from these two nodes for their switch settings. Thirdly, we can see that the most frequently used operation in the distributed algorithms is addition (or addition-like operations). For an $n \times n$ RBN, the maximum values of n_α, n_ϵ, or n_1 are n, which can be represented by using at most $\log n$ bits. We can implement the adder in a pipelined fashion such that the $\log n$ bit adder is reduced to a one bit adder. Also, since the distributed algorithms work in a pipelined fashion, the delay caused by running the forward and the backward processes is also reduced.

Next, we analyze the hardware cost, network depth and set-up time of the new multicast network. Since we add only a constant cost (a constant number of one bit adders) to each switch for the self-routing circuit and there are $\frac{n}{2}\log n$ switches in an $n \times n$ RBN, the hardware cost for the RBN is $O(n \log n)$, and is the same for an $n \times n$ BSN. Let $C(n)$ denote the cost of an $n \times n$ BRSMN. By the recursive construction in Figure 1, we have $C(n) = O(n \log n) + 2C\left(\frac{n}{2}\right)$, which implies $C(n) = O(n \log^2 n)$. Since the network depth of an $n \times n$ RBN is $O(\log n)$, so is an $n \times n$ BSN. Let $D(n)$ denote the network depth for an $n \times n$ BRSMN. From $D(n) = O(\log n) + D\left(\frac{n}{2}\right)$, we immediately have $D(n) = O(\log^2 n)$.

Note that the distributed switch setting algorithms work in a pipelined fashion. It takes $O(\log n)$ unit time delay for the first bit (from input) to reach a switch in the last stage of an $n \times n$ RBN. Then it takes only $O(1)$ unit time for each of the subsequent $\log n - 1$ bits to reach a switch in the last stage. Hence, the propagation delay in the forward phase is $O(\log n)$ unit time, and so is in the backward phase. By a similar analysis, the total propagation delay of the switch setting algorithm for an $n \times n$ BRSMN is $O(\log^2 n)$.

Moreover, note that in our design all major functional components are recursively defined reverse banyan networks. As shown in [11], by reusing part of the network we can construct a feedback version of our multicast network with $O(n \log n)$ cost.

Finally, we can see that the RBN-based multicast network presented in this paper achieves the same order of complexities of network cost, network depth, set-up time as those of the RBN-based permutation network proposed by Cheng and Chen[10].

9 Conclusions

In this paper, we have proposed a design for a new self-routing multicast network based on recursive decompositions of multicast networks. Different from earlier proposed multicast networks, the new multicast network is conceptually simple and has a good modularity. The network design is based on the binary radix sorting concept and all functional components of the network are recursively constructed reverse banyan networks. In addition, the routing circuit is completely distributed into each switch of the network so that the network is operated in a self-routing manner. The new multicast network we design is compared favorably with the previously proposed multicast networks. It uses $O(n \log^2 n)$ logic gates, and has $O(\log^2 n)$ gate delay and $O(\log^2 n)$ set-up time where the unit of time is a gate delay. By re-using part of the network, the feedback version of our design can further reduce the network cost to $O(n \log n)$.

References

[1] J. Duato, S. Yalmanchili, and L.M. Ni, *Interconnection Networks: An Engineering Approach*, IEEE Computer Society Press, Los Alamitos, CA, 1997.

[2] D. Nassimi and S. Sahni, "Parallel permutation and sorting algorithms and a new generalized connection network," *J. ACM*, Vol. 29, No 3, pp. 642-667, 1982.

[3] T.T. Lee, "Nonblocking copy networks for multicast packet switching," *IEEE J. Selected Areas in Comm.*, Vol. 6, No. 9, pp. 1455-1467, 1988.

[4] C.-T. Lea, "A new broadcast switching network," *IEEE Trans. Comm.*, Vol. COM-36, No. 10, pp. 1128-1137, 1988.

[5] Y. Yang and G.M. Masson, "Nonblocking broadcast switching networks," *IEEE Trans. Computers*, Vol. C-40, pp. 1005-1015, 1991.

[6] C. Lee and A.Y. Oruç, "Design of efficient and easily routable generalized connectors," *IEEE Trans. Comm.*, Vol. COM-43, No. 2/3/4, pp. 646-650, 1995.

[7] Y. Yang and G.M. Masson, "Broadcast ring sandwich networks," *IEEE Trans. Computers*, Vol. C-44, No. 10, pp. 1169-1180, 1995.

[8] Y. Yang, "A class of interconnection networks for multicasting," accepted for publication in *IEEE Trans. Computers*.

[9] C.Y. Jan and A.Y. Oruç, "Fast self-routing permutation switching on an asymptotically minimum cost network," *IEEE Trans. Computers*, Vol. 42, No. 12, pp. 1469-1479, 1993.

[10] W.-J. Cheng and W.-T Chen, "A new self-routing permutation network," *IEEE Trans. Computers*, Vol. C-45, No. 5, pp. 630-636, 1996.

[11] Y. Yang and J. Wang, "A new self-routing multicast network," *University of Vermont CS Research Report #98105-01*, 1998.