International Journal of High Speed Electronics, Vol. 3, Nos. 3 & 4 (1992) 279–296

AN MIN-BASED ARCHITECTURE FOR RECONFIGURABLE DE BRUIJN STRUCTURES

S. SRINIVAS*, K. THULASIRAMAN, and M. N. S. SWAMY
Department of Electrical and Computer Engineering, Concordia University, Montreal, Quebec H3G 1M8, Canada

Received 2 October 1992

This paper presents a novel parallel architecture based on a multistage interconnection network (MIN) for reconfigurable binary de Bruijn structures. The proposed architecture is able to assume distinct binary de Bruijn configurations (BDCs), where each configuration has the geometric pattern or structure as that of a binary de Bruijn graph. A system with N nodes or processing elements can generate $N^2/4$ distinct BDCs. The novelty of the architecture is in the design of the switching network for interconnecting the nodes. The switching network adopted is an augmented shuffle-exchange MIN. The favorable features of the architecture include fast reconfiguration, simplified hardware in the MIN, absence of the need for reconfiguration hardware in the nodes, and simple routine control.

The generation of BDCs is derived from an equation, called the Reconfiguration Equation, which is based on simple logical operations and defines the necessary interconnections among the nodes. It is shown that the architecture assumes interconnections according to this equation and consequently the proof of reconfiguration is given. The important properties of the reconfigurable de Bruijn structure are outlined. Finally, two features which are useful in enhancing the reconfigurability of the architecture are discussed. First, it is proved that the architecture can be augmented to generate partitioned de Bruijn configurations. Second, it is shown that the architecture can assume distinct binary tree configurations by a simple modification.

1. Introduction

The de Bruijn network architecture, which is based on the de Bruijn graph,[1] has received much interest of late as a versatile architecture for parallel processing. In particular, the binary de Bruijn structure has been advocated as an effective network architecture for a wide range of applications.[2] The versatility of the binary de Bruijn structure arises from its ability to solve a large class of problems since it can admit important topologies such as the ring, the linear array, the complete binary tree, the tree machine, and the shuffle-exchange network. It also has the advantages of ease in implementation compared to the hypercube since the number of interconnections per node does not grow with the network size. Furthermore,

*Present address: Division of Computing, Department of Mathematics, Statistics, and Computing Science, Dalhousie University, Halifax, Nova Scotia B3H 3J5, Canada.

it exhibits fairly good fault-tolerance properties. Many fault-tolerance strategies and routing algorithms for de Bruijn networks in general, and binary de Bruijn networks in particular, have been proposed.[2–8] NASA's Galileo project (cited in Ref. 2), which aims at building a massively parallel de Bruijn network architecture, indicates the importance of the network in the industrial sector.

One of the important strategies in the design of high-performance systems is the use of reconfigurable parallel architectures. Such an architecture can speed up algorithm execution, minimize interprocessor communication delays, improve resource utilization, and provide for enhanced fault-tolerance. Many reconfigurable parallel systems of a different philosophy, reconfiguration technique, and application scope have been proposed.[9–16]

The work on de Bruijn networks reported thus far considers the network to be static. The main contribution of this paper is to propose a novel architecture for reconfigurable binary de Bruijn structures. The architecture is capable of assuming distinct binary de Bruijn configurations (BDCs) among its nodes or processing elements, where each configuration has the same geometric pattern of the binary de Bruijn graph. The number of distinct BDCs possible is $N^2/4$, where N is the number of nodes in the system.

The novelty of the architecture is in the design of the switching network for interconnecting the nodes. The switching network adopted is an augmented shuffle-exchange multistage interconnection network (MIN). The favorable features of the architecture include fast reconfiguration, simplified hardware in the MIN, absence of the need for reconfiguration hardware in the nodes, and simple routing control.

The idea proposed in this paper has its basis in the work of Biswas and Srinivas,[18] which reports a reconfigurable architecture for binary tree structures. The architecture with N nodes is capable of assuming N distinct binary tree configurations (BTCs). As mentioned earlier, we define a binary de Bruijn configuration as having the geometric pattern of a binary de Bruijn graph. Based on this definition, we utilize the idea that a BDC can be obtained by a concatenation of two edge-disjoint binary tree configurations. In the light of this, we augment the binary tree architecture to generate distinct BDCs.

The generation of BDCs is derived from an equation, which we call the Reconfiguration Equation, which is based on simple logical operations and defines the interconnections among the nodes. We show that the architecture assumes interconnections according to this equation and consequently prove that it is capable of reconfiguration. We enumerate the important properties of the reconfigurable binary de Bruijn structure. Finally, we discuss two features which are useful in enhancing the reconfigurability of the proposed architecture. First, we prove that the architecture can be augmented to generate partitioned binary de Bruijn configurations. Specifically, the architecture can assume distinct pairs of independent binary de Bruijn configurations. Second, we show that by a simple modification, the architecture can function as the reconfigurable binary tree architecture of Biswas and Srinivas.[18]

The rest of the paper is outlined as follows. In the next section, the concept of a reconfigurable binary de Bruijn structure is introduced. Section 3 briefly outlines the important features of the binary tree architecture in Ref. 18. Section 4 describes the proposed reconfigurable architecture, gives the proof for reconfiguration, and enumerates the properties of the reconfigurable de Bruijn structure. In Sec. 5, the architecture is augmented to generate partitioned de Bruijn configurations. In Sec. 6, we show that a simple modification enables the architecture to generate distinct binary tree configurations. Section 7 provides concluding remarks and enumerates the advantages of the proposed architecture.

2. The Reconfigurable Binary de Bruijn Structure

Generally, a *node* in a parallel architecture refers to a *processing element*, which consists of a processor unit and a memory unit. We use the term *structure* of a parallel architecture to refer to its topology or the geometric pattern formed by the nodes and their interconnecting links. The term *configuration* refers to the structure in which the location of each node has been specified. For instance, we may have different configurations of the binary tree structure, each with a different root node. *Reconfiguration* refers to the process of altering the configuration of an architecture by rearranging the interconnecting paths. A *reconfigurable structure* is one that can assume distinct configurations. The proposed architecture for reconfigurable de Bruijn structures is capable of assuming distinct binary de Bruijn configurations.

The number of nodes N in the architecture is assumed to be 2^k, where k is a positive integer. Each node is denoted as $P(i)$, $i = 0, 1, \ldots, N-1$, and can be represented as a k-bit (binary) number $i_{k-1}i_{k-2}\cdots i_0$ which gives the binary value of the index i of $P(i)$. For the sake of simplicity, we shall often refer to $P(i)$ by its decimal index i only.

Definition 1: A *binary tree structure* (BTS) with 2^k nodes is that having $k+1$ levels $(L_0, L_1, \ldots, L_k)$ of nodes with the *root* at L_0 and the *leaves* at L_k. Each node at an *intermediate level* L_x, $x = 1, 2, \ldots, k-1$, connects to two nodes at L_{x+1}, while the root connects to one node at L_1 and has one connection with itself.

Figure 1(a) shows the BTS with eight nodes. Figure 1(b) shows two possible configurations of the structure that can be generated by the reconfigurable binary tree architecture proposed in Ref. 18.

Definition 2: Consider a node $P(l)$ at level L_x, $x = 1, 2, \ldots, k-1$, in a BTS. Let the two nodes to which it is connected at L_{x+1} be $P(m)$ and $P(n)$. Then $P(m)$ and $P(n)$ are the *predecessors* of $P(l)$. $P(l)$ is called the *successor* of $P(m)$ (or $P(n)$). $P(m)$ ($P(n)$) is called the *sibling* of $P(n)$ ($P(m)$). The pair of nodes ($P(m)$, $P(n)$) forms a *sibling pair*.

Extending the above definition to the root, it can be easily seen that one of the predecessors of the root is the node at L_1; the other predecessor is the root

itself. Furthermore, the root and the node at L_1 form a sibling pair. As an example to illustrate the above definition, consider the node 6 in the first configuration in Fig. 1(b). Its predecessors are 7 and 5. The pair of nodes (7,5) forms a sibling pair. Similarly, (3,1), (6,4), and (2,0) each forms a sibling pair.

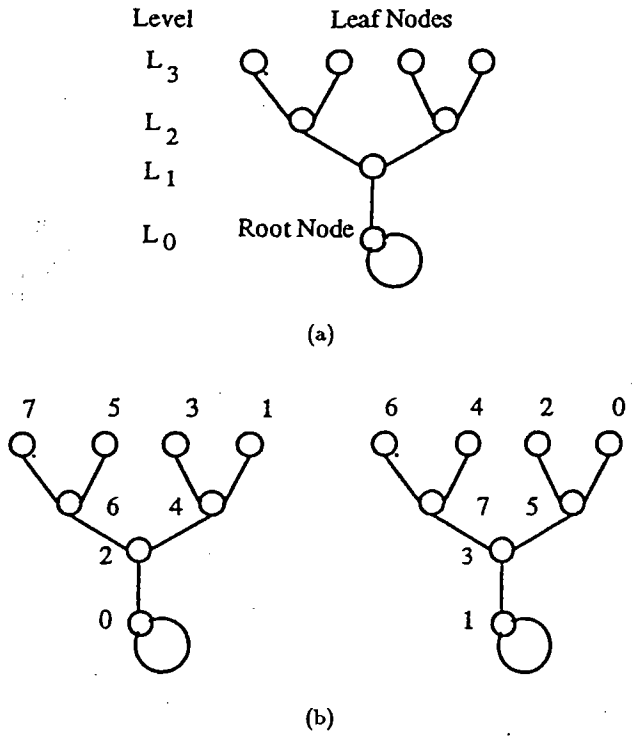


Fig. 1. (a) A binary tree structure with eight nodes, and (b) two configurations of the structure.

Definition 3: An undirected binary de Bruijn graph (BDG) is that in which any node $P(i) = i_{k-1}i_{k-2}\cdots i_0$ has four neighbors given by[a] $i_{k-2}i_{k-3}\cdots i_1 i_0 i_{k-1}$, $i_0 i_{k-1} i_{k-2}\cdots i_2 i_1$, $i_{k-2}i_{k-3}\cdots i_1 i_0 \bar{i}_{k-1}$, and $\bar{i}_0 i_{k-1} i_{k-2}\cdots i_2 i_1$.

Figure 2(a) shows the undirected BDG with eight nodes. It can be easily verified[17] that the graph can be drawn as a concatenation of two BTCs (Fig. 2(b)). The component BTCs of the graph are edge-disjoint and *leaf-set disjoint*, that is, the set of leaf nodes in the two BTCs are disjoint. For instance, in Fig. 2(b), one of the component BTCs has 0 as the root and nodes 4, 5, 6, and 7 as leaves, while the other has 7 as the root and 0, 1, 2, and 3 as leaves.

We generalize the above concept and define a binary de Bruijn structure (BDS) to be a concatenation of two binary tree structures. The BDS has the same topology

[a] $\bar{a}$ represents the complement of the binary variable a.

as that of the undirected BDG. But the only difference is that the node locations are not specified. The following is a formal definition.

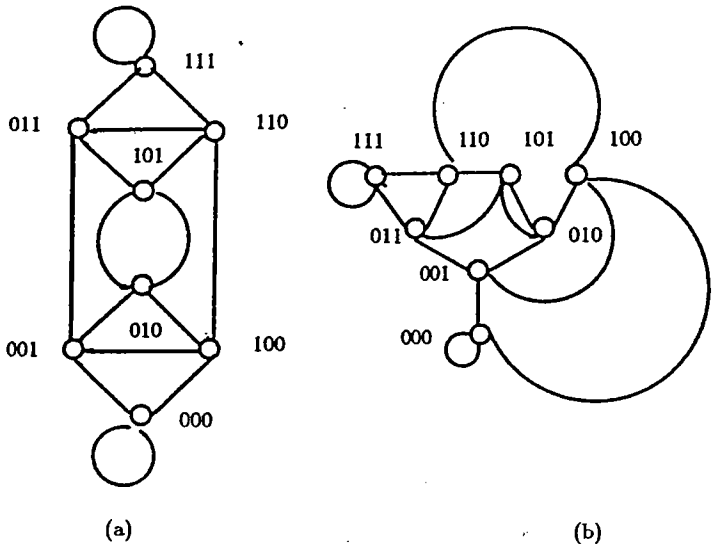


Fig. 2. (a) An undirected binary de Bruijn graph with eight nodes, and (b) its equivalent representation.

Definition 4: A *binary de Bruijn structure* with 2^k nodes is that formed by the concatenation of two edge-disjoint and leaf-set disjoint $(k+1)$-level binary tree structures. A *binary de Bruijn configuration* refers to the BDS in which the location of each node has been specified.

Figure 3 shows two possible configurations of the binary de Bruijn structure with eight nodes that can be obtained by the architecture proposed in this paper. Consider, for example, the first configuration of Fig. 3(a). It is composed of two BTCs, one with 0 as the root and 7, 5, 3, and 1 as leaf nodes; while the second BTC has 7 as the root and 6, 4, 2, and 0 as the leaf nodes.

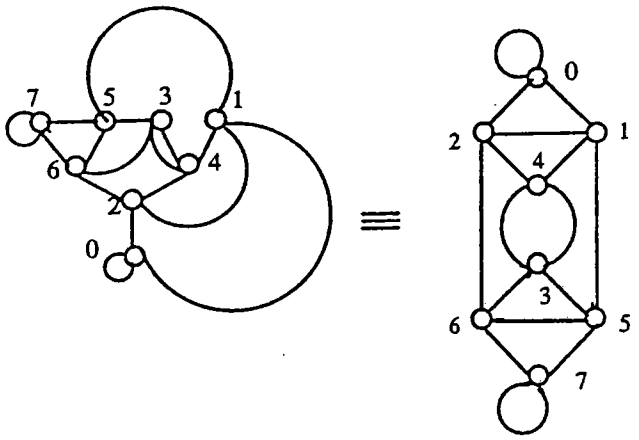
3. The Reconfigurable Binary Tree Architecture

The reconfigurable binary tree architecture[18] with $N = 2^k$ nodes is capable of assuming N distinct BTCs and is implemented with a k-stage shuffle-exchange MIN. The existing shuffle-exchange MIN[19] does not have the capability of assuming a BTC among the nodes. This capability has been imparted to the network by a simple augmentation. The reconfiguration in the architecture is based on the following equation, which gives the relation between each node $P(i)$ and the node $P(j)$ to which it gets connected for each value of a k-bit control code $C = c_{k-1}c_{k-2}\cdots c_0$

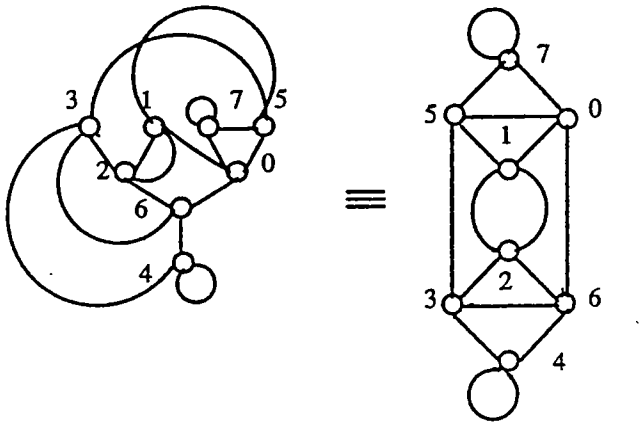
controlling the switching in the MIN

$$P(j) = (CRS[P(i)] \Lambda B) \oplus C \tag{1}$$

where $CRS[P(i)]$ is a one-bit circular right shift of the binary representation of $P(i)$, B is a k-bit number with value $b_{k-1}b_{k-2}\cdots b_1 b_0 = 11\cdots 10$, Λ is the bitwise AND, and $\oplus$ is the bitwise EXOR operator. It has been proved in Ref. 18 that an architecture which establishes interconnections among its nodes according to (1) is able to generate distinct BTCs, one for each value of C. Figure 4 shows the BTCs in an eight-node architecture.



(a)



(b)

Fig. 3. Two possible binary de Bruijn configurations with eight nodes.

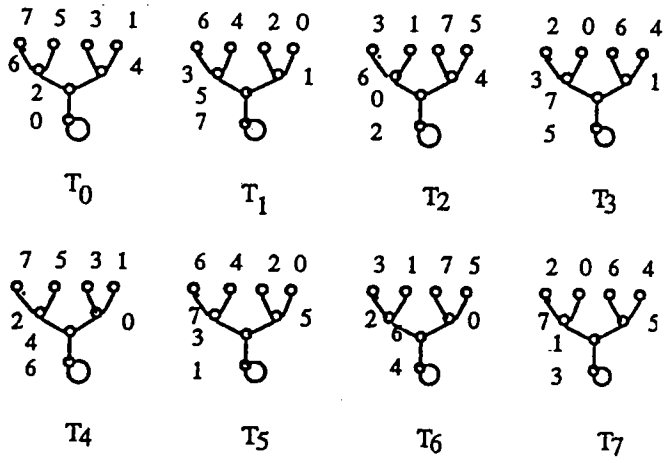


Fig. 4. The eight distinct BTCs generated for different values of C in an 8-node binary tree architecture.

Notation 1: Let D be the decimal equivalent of the control code $c_{k-1}c_{k-2}\cdots c_0$ generating a BTC. We shall denote the configuration as T_D or $T_{c_{k-1}c_{k-2}\cdots c_0}$.

Property 1: The same pair of nodes form a sibling pair in all the BTCs.

For example, it can be observed in Fig. 4 that the nodes 2 and 0 form a sibling pair in all the configurations. So do the pairs (1,3), (6,4), and (7,5).

Property 2: A node $P(j) = j_{k-1}j_{k-2}\cdots j_0$ is a leaf node if and only if $j_0 = \bar{c}_0$.

This gives an important result.

Lemma 1: If D is an even number (including zero), then T_D has odd-numbered nodes as leaves; if D is odd, then T_D has even-numbered nodes as leaves.

For example, T_0, T_2, T_4, and T_6 have 7, 5, 3, and 1 as leaf nodes. On the other hand, T_1, T_3, T_5, and T_7 have 0, 2, 4, and 6 as leaf nodes.

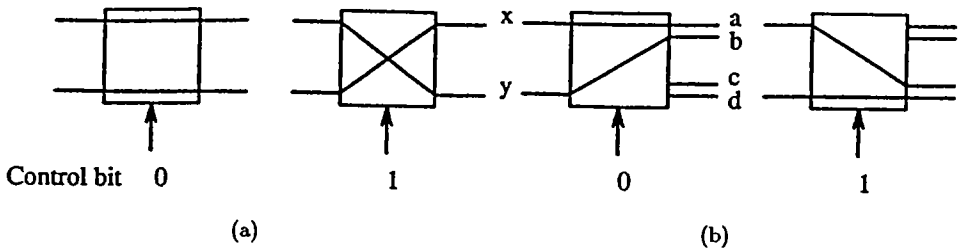


Fig. 5. Switching states of (a) a basic switching element and (b) an augmented basic switching element.

We now define two switching elements that are used in the MIN of the tree architecture. Figure 5(a) depicts a basic switching element (BSE). It is a 2×2 switch with two switching states: straight (if the control bit = 0), and exchange (if the control bit = 1). Figure 5(b) shows an augmented basic switching element (ABSE). It is a 2×4 switch with two states: if the control bit = 0 (1), then the inputs x and y connect to the upper (lower) outputs a and b (c and d), respectively.

4. The Proposed Architecture

The $N = 2^k$ nodes in the architecture are interconnected by an MIN of k stages, which are numbered $S_{k-1}, S_{k-2}, \ldots, S_0$ in order, with S_{k-1} being the leftmost stage. Each stage S_p, $p = k-1, k-2, \ldots, 0$, consists of $N/2$ switching elements (SEs) and is controlled by two controls $c_{1,p}c_{2,p}$. S_p, $p = k-1, k-2, \ldots, 1$, is connected to S_{p-1} by the perfect shuffle connection.

We now describe the switching states of the SEs. Each SE in any stage S_p, $p = k-1, k-2, \ldots, 1$, is a 4×4 switch (Fig. 6(a)). Let us label its inputs as IT1, IT2, IB1, and IB2, and its outputs as OT1, OT2, OB1, and OB2. In this notation, I (O) represents input (output), T (B) denotes top (bottom) terminal, and 1 or 2 its serial number. Figures 6(b) and (c) depict the switching states of the SE for different values of the bits $c_{1,p}c_{2,p}$. It can be observed that the SE is logically equivalent to a concatenation of two BSEs, namely, BSE1 composed of lines (IT1, IB1, OT1, OB1) and BSE2 (IT2, IB2, OT2, OB2) and controlled by $c_{1,p}$ and $c_{2,p}$, respectively. If $c_{1,p}$ is 0 (1), then BSE1 is set to the straight (exchange) state. Similarly, BSE2 is set to the straight (exchange) state when $c_{2,p}$ is 0 (1).

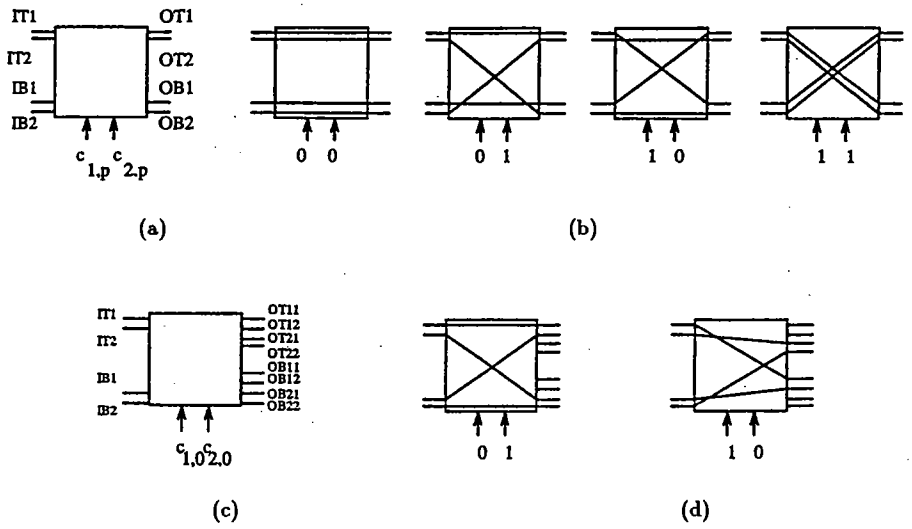


Fig. 6. (a) Structure and (b) switching states of an SE in stage S_p, $p = k-1, k-2, \ldots, 1$; (c) structure and (d) switching states of an SE in S_0.

The switching states of an SE in S_0 are different. Figure 6(c) shows the inputs, outputs, and control lines of the SE. It is a 4×8 switch and is effectively a concatenation of two ABSEs, namely, ABSE1 composed of lines (IT1, IB1, OT11, OT12, OB11, OB12) and ABSE2 (IT2, IB2, OT21, OT22, OB21, OB22) and controlled by $c_{1,0}$ and $c_{2,0}$, respectively. Figure 6(d) depicts the switching states. The bits $c_{1,0}$ and $c_{2,0}$ are always complements of each other. If $c_{1,0} = 0$ and $c_{2,0} = 1$, then the inputs IT1 and IB1 connect to OT11 and OT12, respectively, while IT2 and IB2 connect to OB21 and OB22, respectively. If $c_{1,0} = 1$ and $c_{2,0} = 0$, then IT1 and IB1 connect to OB11 and OB12, respectively, while IT2 and IB2 connect to OT21 and OT22, respectively.

All the SEs of a particular stage receive the same control bits and hence switch to the same state. Thus, the entire MIN is controlled by a $2k$-bit *control code* $C = c_{1,k-1}c_{2,k-1}\ c_{1,k-2}c_{2,k-2} \cdots c_{1,0}c_{2,0}$. The control code is issued by a central Configuration Controller (CC).

Figure 7 illustrates the architecture for $N = 8$ with switching states shown for $C = 10\ 10\ 01$. An observation of the figure indicates the manner in which the nodes are connected to the MIN. Each node has six terminals, namely, t_{11}, t_{12}, t_{13}, t_{21}, t_{22}, and t_{23}. Consider the node $P(0)$. Its terminals are connected to the top SE in S_0 in the following manner: $t_{11} \rightarrow$ OT11, $t_{12} \rightarrow$ OT12, $t_{21} \rightarrow$ OT21, $t_{22} \rightarrow$ OT22.

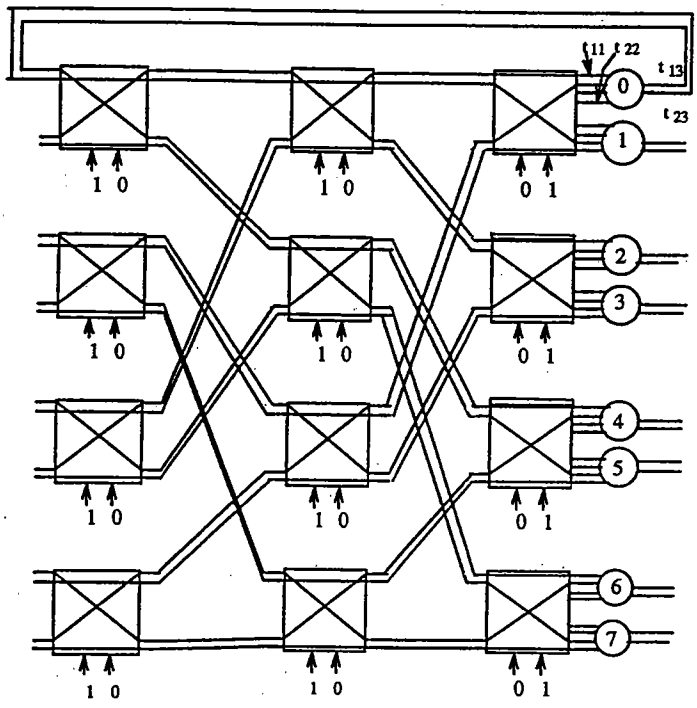


Fig. 7. The proposed architecture for reconfigurable binary de Bruijn structures.

The connections from t_{13} and t_{23} are brought out to the front of the MIN and connected to IT1 and IT2, respectively, of the top SE in S_{k-1}. The terminals of $P(1)$ are similarly connected to the bottom four outputs of the top SE in S_0 and the bottom two terminals of the top SE in S_{k-1}. Note that even (odd) numbered nodes are connected to the top (bottom) terminals of an SE.

With this architecture, we show that it is possible to obtain distinct binary de Bruijn configurations among the nodes. The configurations are stage-controlled and synchronous. For each value of the control code issued by the CC, a BDC is obtained. This property follows from an equation given below.

4.1. *Reconfiguration equation for binary de Bruijn structures*

Theorem 1: Let $C1 = c_{k-1}c_{k-2}\cdots c_1c_0$ and $C2 = c'_{k-1}c'_{k-2}\cdots c'_1c'_0$ be two distinct k-bit variables such that $c_p, c'_p \in \{0,1\}$ for $p = k-1, k-2, \ldots, 0$, and $c_0 = \bar{c}'_0$. Let each node $P(i)$ establish connection with two nodes $P(j1)$ and $P(j2)$ such that

$$\left.\begin{aligned} P(j1) &= (CRS[P(i)] \Lambda B) \oplus C1 \\ P(j2) &= (CRS[P(i)] \Lambda B) \oplus C2 \end{aligned}\right\} \tag{2}$$

where $CRS[P(i)]$ represents a one-bit circular right shift of $P(i)$ and $B = b_{k-1}b_{k-2}\cdots b_1b_0 = 11\cdots10$. Then a binary de Bruijn configuration is established among the nodes for any arbitrary value of $C1$ and $C2$.

Proof: It is known from Sec. 3 that if the nodes are interconnected according to (1), then a BTC is obtained. In the present case, the configuration is determined by (2), which consists of two equations for $P(j1)$ and $P(j2)$, each of which is identical to (1). Hence the configuration is logically equal to two BTCs, say, T_x and T_y, generated by the control codes $C1$ and $C2$, respectively. Since $c_0 = \bar{c}'_0$, hence it follows from Lemma 1 that T_x and T_y are leaf-set disjoint.

Now draw T_x and T_y in concatenation. First draw T_x and then add the edges of T_y. Since T_x and T_y are leaf-set disjoint, hence the root of T_y is a leaf node of T_x. This node is connected to its sibling. Every other leaf node of T_x is connected to two nodes, which form a sibling pair. From Property 1 in Sec. 3, it follows that every leaf node of T_x is connected to a distinct sibling pair. Hence the concatenation of T_x and T_y is a binary de Bruijn configuration.

Example 1: Consider an architecture with eight nodes and let the nodes establish connections according to Theorem 1 with $C1 = 000$ and $C2 = 001$. We can determine the various interconnections among the nodes using (2) as follows:

$P(i)$	0	1	2	3	4	5	6	7
$P(j1)$	0	4	0	4	2	6	2	6
$P(j2)$	1	5	1	5	3	7	3	7

It is seen that this corresponds to the BDC of Fig. 3(a).

Equation (2) is called the Reconfiguration Equation for binary de Bruijn structures. It is significant in that it forms the basis for the proof that the architecture generates distinct BDCs. In the next section, we show that the proposed architecture establishes interconnections according to this equation, and consequently prove that it is capable of reconfiguration.

4.2. *Proof of reconfiguration*

Theorem 2: For each value of $C = c_{1,k-1}c_{2,k-1}c_{1,k-2}c_{2,k-2}\cdots c_{1,0}c_{2,0}$, the t_{13} and t_{23} terminals of each node $P(i)$ establish connection through the MIN with two nodes $P(j1)$ and $P(j2)$, which are given by (2), where

$$C1 = c_{1,k-1}c_{1,k-2}\cdots c_{1,0} \tag{3}$$

$$C2 = c_{2,k-1}c_{2,k-2}\cdots c_{2,0}\ . \tag{4}$$

Proof: Can be obtained by starting at the t_{13} and t_{23} terminals of an arbitrary node $P(i)$ and tracing the paths through the MIN to the nodes to which they are connected.

Example 2: In Fig. 7, $C1 = 110$ and $C2 = 001$. Consider the node $P(0) = 000$. Its terminals t_{13} and t_{23} are connected, respectively, to $(CRS[000]\Lambda 110) \oplus 110 = 110(P(6))$ and $(CRS[000]\Lambda 110) \oplus 001 = 001(P(1))$. This can be verified by tracing the paths from $P(0)$ through the MIN.

Theorem 3: A binary de Bruijn configuration is established for each value of C.

Proof: Follows from Theorem 1 and Theorem 2.

We see from the above discussion that the BDC-generated with $N = 2^k$ nodes is a concatenation of two BTCs, T_x and T_y, such that $x, y \in \{0, 1, \ldots, N-1\}$ and that if x is odd, y is even, and if y is odd, then x is even. We denote the BDC as T_xT_y.

Theorem 4: The architecture can generate $N^2/4$ distinct BDCs.

Proof: C is a $2k$-bit string with the constraint that $c_{1,0} = \bar{c}_{2,0}$. Hence, it can take 2^{2k-1} values. Therefore, the total number of BDCs is $2^{2k-1} = N^2/2$. A BDC T_xT_y is identical to the BDC T_yT_x. Hence, the total number of distinct BDCs is $(N^2/2)/2 = N^2/4$.

Example 3: With $N = 8$, the architecture can generate sixteen BDCs, namely, T_0T_1, T_0T_3, T_0T_5, T_0T_7, T_1T_2, T_1T_4, T_1T_6, T_2T_3, T_2T_5, T_2T_7, T_3T_4, T_3T_6, T_4T_5, T_4T_7, T_5T_6, and T_6T_7. Figure 8 shows all the sixteen configurations. The BDC that is generated corresponding to the switching states of Fig. 7 is T_1T_6. Note that T_1T_6 is same as T_6T_1 and the control code generating the BDC is 10 10 01 where $C1 = 110$ and $C2 = 001$. The control code $C = 01\ 01\ 10$ would also have generated

the same BDC. Similarly, another BDC T_0T_5 (for instance) is generated by any of the two control codes 01 00 01 or 10 00 10.

4.3. *Properties of the reconfigurable de Bruijn structure*

In what follows, we derive some useful properties of the reconfigurable binary de Bruijn structure.

Theorem 5: (Neighbors of a node) Given a control code C and a node $P(i)$, the four neighbors of $P(i)$ in the BDC T_xT_y that is generated can be determined as follows: If $i_0 = c_{1,0}$, then the neighbors of $P(i)$ are $P(j1)$, $P(j2)$, $P(m1)$, and $P(n1)$; if $i_0 = c_{2,0}$, then the neighbors of $P(i)$ are $P(j1)$, $P(j2)$, $P(m2)$, and $P(n2)$, where

$$P(j1) = i_0 i_{k-1} i_{k-2} \cdots i_2 0 \oplus c_{1,k-1} c_{1,k-2} \cdots c_{1,1} c_{1,0} \tag{5}$$

$$P(j2) = i_0 i_{k-1} i_{k-2} \cdots i_2 0 \oplus c_{2,k-1} c_{2,k-2} \cdots c_{2,1} c_{2,0} \tag{6}$$

$$P(m1) = i_{k-2} i_{k-3} \cdots i_1 0 i_{k-1} \oplus c_{1,k-2} c_{1,k-3} \cdots c_{1,1} 0 c_{1,k-1} \tag{7}$$

$$P(n1) = i_{k-2} i_{k-3} \cdots i_1 1 i_{k-1} \oplus c_{1,k-2} c_{1,k-3} \cdots c_{1,1} 0 c_{1,k-1} \tag{8}$$

$$P(m2) = i_{k-2} i_{k-3} \cdots i_1 0 i_{k-1} \oplus c_{2,k-2} c_{2,k-3} \cdots c_{2,1} 0 c_{2,k-1} \tag{9}$$

$$P(m2) = i_{k-2} i_{k-3} \cdots i_1 1 i_{k-1} \oplus c_{2,k-2} c_{2,k-3} \cdots c_{2,1} 0 c_{2,k-1} \,. \tag{10}$$

Proof: The successors and predecessors of $P(i)$ in T_x and T_y are its neighbors. From (1), the successor $P(j1)$ of $P(i)$ in the BTC T_x is given by (5). If $i_0 = c_{1,0}$, then $P(i)$ has two predecessors $P(m1)$ and $P(n1)$ in T_x. Again, from (1), these two nodes are given by (7) and (8), respectively. If $i_0 = \bar{c}_{1,0} = c_{2,0}$, then $P(i)$ has no predecessors (it is a leaf node in T_x). Similarly, the successor $P(j2)$ of $P(i)$ and its predecessors $P(m2)$ and $P(n2)$ (if they exist) in T_y are given by (6), (9), and (10), respectively.

Example 4: Determine the neighbors of the node 100 in the BDC generated by the control code 01 10 10: In this case, $i_0 = \bar{c}_{2,0}$. Hence the neighbors are $(010 \oplus 011) = 001$, $(010 \oplus 100) = 110$, $(001 \oplus 001) = 000$, and $(011 \oplus 001) = 010$. This can be verified in Fig. 8.

Theorem 6: (Control codes for adjacency) Given any two nodes $P(i) = i_{k-1} i_{k-2} \cdots i_0$ and $P(j) = j_{k-1} j_{k-2} \cdots j_0$, determine the control bits given by the following

$$c_{k-1} c_{k-2} \cdots c_1 c_0 = i_0 i_{k-1} i_{k-2} \cdots i_2 0 \oplus j_{k-1} j_{k-2} \cdots j_0 \tag{11}$$

$$c'_{k-1} c'_{k-2} \cdots c'_1 c'_0 = j_0 j_{k-1} j_{k-2} \cdots j_2 0 \oplus i_{k-1} i_{k-2} \cdots i_0 \,. \tag{12}$$

Then any control code of the form $c_{k-1}d\,c_{k-2}d\cdots c_1 d\,c_0\bar{c}_0$, $dc_{k-1}\,dc_{k-2}\cdots dc_1\,\bar{c}_0 c_0$, $c'_{k-1}d\,c'_{k-2}d\cdots c'_1 d\,c'_0\bar{c}'_0$, or $dc'_{k-1}\,dc'_{k-2}\cdots dc'_1\,\bar{c}'_0 c'_0$, where $d \in \{0,1\}$, will generate a BDC in which $P(i)$ and $P(j)$ are adjacent to each other.

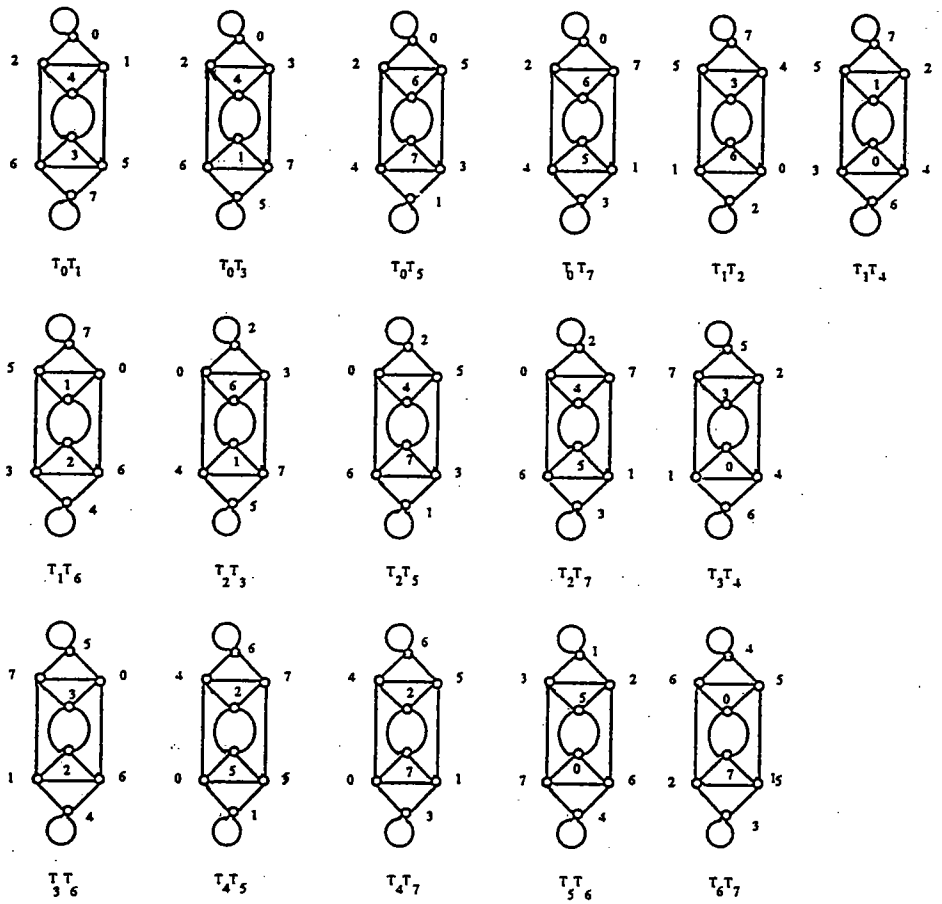


Fig. 8. The sixteen distinct BDCs obtained with the architecture of Fig. 7.

Proof: Without loss of generality, consider the architecture with eight nodes. If $P(i)$ and $P(j)$ are adjacent in a BDC T_xT_y, then they are adjacent in either T_x or T_y. In any BTC, $P(i)$ and $P(j)$ are adjacent if $P(i)$ is the successor of $P(j)$ or if $P(j)$ is the successor of $P(i)$. Let $C = c_2c_1c_0$ be the control code generating the BTC which makes $P(j)$ the successor of $P(i)$. Hence, from (1),

$$c_2c_1c_0 = (CRS[P(i)]\Lambda B) \oplus P(j)\ . \tag{13}$$

Similarly, let $C' = c'_2c'_1c'_0$ be the control code generating the BTC which makes $P(i)$

the successor of $P(j)$. Again, from (1),

$$c'_2c'_1c'_0 = (CRS[P(j)\Lambda B) \oplus P(i) \; . \tag{14}$$

Hence the two nodes are adjacent in the set of BDCs $\{T_{c_2c_1c_0}T_{dd\bar{c}_0}, T_{dd\bar{c}_0}T_{c_2c_1c_0}, T_{c'_2c'_1c'_0}T_{dd\bar{c}'_0}, T_{dd\bar{c}'_0}T_{c'_2c'_1c'_0}\}$. The theorem generalizes this result.

Example 5: Find the control codes which generate BDCs in which the nodes 001 and 011 are adjacent to each other: In this case, $c_2c_1c_0 = (100 \oplus 011) = 111$ and $c'_2c'_1c'_0 = (100 \oplus 001) = 101$. Then the required set of control codes is $\{1d1d10, d1d101, 1d0d10, d1d001 : d \in \{0,1\}\}$.

It can be inferred from the above discussion that, given any pair of nodes, we can always generate a BDC in which these two nodes are adjacent to each other. This property is useful in that any two nodes between which message traffic is heavy can be brought adjacent to each other. Furthermore, fault-tolerance of the system can be obtained. Suppose that in a BDC, two nodes $P(m)$ and $P(n)$ are communicating with each other along a path which contains one intermediate node, say $P(l)$. Suppose that $P(l)$ goes faulty. Now we can reconfigure the system into another BDC in which $P(m)$ and $P(n)$ are adjacent to each other, and the faulty node is bypassed.

Embedding other networks: Samatham and Pradhan[2] have demonstrated that the binary de Bruijn graph admits many computationally important networks such as the ring, the linear array, complete binary trees, tree machines, and shuffle-exchange networks. Since the reconfigurable de Bruijn structure proposed in this paper has the same topology as the binary de Bruijn graph, it is intuitive that each BDC generated by the architecture can admit all of the above networks. For instance, the binary de Bruijn graph with N nodes can admit at least four complete binary trees of height $\log_2 N$.[2] (Note that here the complete binary tree has the usual graph-theoretic definition.) Hence, in our case, each BDC can admit at least four such trees. Hence the total number of possible complete binary trees is $N^2/4 \times 4 = N^2$. Similarly, many distinct configurations of different networks enhances the message-routing efficiency and the fault-tolerance capabilities of the architecture.

5. Augmentation of the Architecture for Partitioning

We now show that a simple augmentation to the proposed architecture enable it to generate partitioned binary de Bruijn configurations. Specifically, we show that distinct pairs of partitioned BDCs can be obtained by augmenting the last stage S_0. Suppose that another control line c_P is added to the last stage S_0. Normally, $c_P = 0$, and all the SEs in S_0 have the same switching state. If $c_P = 1$, then the switching in S_0 is designed such that the SEs in the upper half of S_0 is controlled by $c_{1,0}c_{2,0}$, while the SEs in the lower half are controlled by $\bar{c}_{1,0}\bar{c}_{2,0}$. The following interesting result was observed with this augmentation.

Theorem 7: A control code C in which $c_{1,k-1}c_{2,k-1} = c_{1,0}c_{2,0} = a\bar{a}$ for any $a \in \{0,1\}$ generates two independent BDCs.

Proof: Without loss of generality, let $N = 8$. Tracing the paths through the MIN from t_{13} of an arbitrary node $P(i)$ to its connecting node $P(j1)$, we get

$$P(j1) = \begin{cases} (i_0 \oplus a)(i_2 \oplus c_{1,1})\, a & \text{if } i_0 \oplus a = 0 \\ (i_0 \oplus a)(i_2 \oplus c_{1,1})\, \bar{a} & \text{if } i_0 \oplus a = 1 \end{cases}$$

which can be rewritten as

$$P(j1) = (i_0 \oplus a)(i_2 \oplus c_{1,1})(i_0)\ . \tag{15}$$

The above equation defines a pair of BTCs since the set of nodes with $i_0 = 0$ are configured by the control code $c_{1,2}c_{1,1}0$, while the set of nodes with $i_0 = 1$ are configured by the control code $c_{1,2}c_{1,1}1$. Similarly, t_{23} of $P(i)$ is connected to

$$P(j2) = (i_0 \oplus \bar{a})(i_2 \oplus c_{2,1})(i_0)\ . \tag{16}$$

The above equation again defines two independent BTCs, which are distinct from the BTCs given by (15). Using the same reasoning as in the proof of Theorem 1, we can prove that the two equations (15) and (16) together define a pair of independent BDCs, in which the nodes with $i_0 = 0$ are configured by the control code $a\bar{a}c_{1,1}c_{2,1}a\bar{a}$, while the set of nodes with $i_0 = 1$ are configured by the control code $\bar{a}ac_{1,1}c_{2,1}\bar{a}a$.

Example 6: With $N = 8$, let $c_{1,2}c_{2,2}\, c_{1,1}c_{2,1}\, c_P c_{1,0}c_{2,0} = 01\ 00\ 101$. Then the two independent BDCs that are generated are shown in Fig. 9.

Thus, the above augmentation enables *partitioning*, that is, the architecture can generate distinct pairs of BDCs. The method can be generalized to obtain a larger number of partitions by augmenting other stages too.

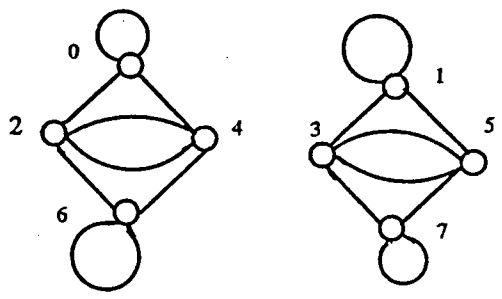


Fig. 9. The partitioned BDCs obtained with the augmentation.

6. Generation of BTCs

As mentioned earlier, the basis for the proposed architecture is the reconfigurable binary tree architecture of Ref. 18. We can convert the proposed architecture to function as the binary tree architecture by a simple modification. Suppose that a control line $c_{s,p}$ is added to each stage S_p, $p = k-1, k-2, \ldots, 0$. Normally, $c_{s,p} = 1$ and the switching states of the SEs are unaltered. If $c_{s,p} = 0$ in any stage S_p, then each SE in that stage functions as a single BSE (single ABSE in S_0). The two switching states can be obtained by using $c_{1,p}$. It is intuitive to see that if $c_{s,p} = 0$ in all the stages, the architecture is equivalent to the binary tree architecture and generates a distinct BTC for each value of $c_{1,k-1}c_{1,k-2}\cdots c_{1,0}$.

Another interesting feature that can be noted is with regard to the generation of partitioned binary tree configurations. By following a strategy similar to Sec. 5, that is, by partitioning the last stage S_0 in this modified architecture, distinct pairs of partitioned BTCs can be generated. The proof for partitioning can be reasoned in a manner similar to Theorem 7.

7. Conclusions

The binary de Bruijn structure is an important computation topology for parallel processing. The work on de Bruijn networks reported thus far considers the network to be static. This paper proposed a novel parallel architecture for reconfigurable binary de Bruijn structures. The architecture with N nodes is capable of assuming $N^2/4$ distinct binary de Bruijn configurations, where each configuration has the geometric pattern of the binary de Bruijn graph.

The architecture is based on a multistage interconnection network. The novelty of the architecture is in the design of the switching elements of the MIN. Reconfiguration is achieved by issuing a single control code to the MIN. In what follows, we outline some of the advantages of the architecture.

The primary advantage of the proposed method is in the fast switching from one configuration to another. Reconfiguration is achieved by controlling the MIN and all the control code bits are broadcast simultaneously. Thus, the direct paths are established in parallel rather than stage by stage. Furthermore, the CC establishes conflict-free paths. These two factors make the system faster than existing methods which employ sequential switching in the MIN.

The architecture does not require separate hardware in the nodes for achieving reconfiguration. Furthermore, the hardware logic for generating control signals in all the switching elements is eliminated. This results in a substantial reduction in hardware in the nodes and the MIN.

No algorithm is required to achieve reconfiguration. Since each control code establishes direct paths between the nodes, no conflict resolution mechanism is required. Furthermore, since the reconfiguration is stage-controlled and synchronous, it results in simple routing and low synchronization overhead.

The capability of having a number of distinct BDCs enhances the performance of the system. For instance, it has been shown that given a pair of nodes, we can always obtain a BDC such that these two nodes are adjacent to each other. This can be used to advantage in decreasing the interprocessor communication delays in the system. Furthermore, such a capability enhances the fault-tolerance inasmuch as restoring the connectivity between two nodes in the event of a faulty node in the path between the two nodes.

We have also shown that the architecture can be augmented to generate partitioned de Bruijn configurations. Furthermore, a simple modification to the architecture enables it to function as a reconfigurable binary tree architecture and generate distinct BTCs. These two types of reconfiguration are useful in further enhancing the performance of the system.

References

1. N. G. de Bruijn, "A combinatorial problem", Koninklijke Netherlands, Academe Van Wetenschapper, Proc. **49** (1946) 758–764.
2. M. R. Samatham and D. K. Pradhan, "The de Bruijn multiprocessor network: A versatile parallel processing and sorting network for VLSI", *IEEE Trans. Comput.* **38** (1989) 567–581.
3. M. L. Schlumberger, "de Bruijn communications networks", Ph.D. dissertation, Stanford University, 1974.
4. M. Imase and M. Itoh, "Design to minimize diameter on building block networks", *IEEE Trans. Comput.* **C-30** (1981) 439–443.
5. D. K. Pradhan and S. M. Reddy, "A fault-tolerant communication architecture for distributed systems", *IEEE Trans. Comput.* **C-31** (1982) 863–870.
6. A. Esfahanaian and S. L. Hakimi, "Fault-tolerant routing in de Bruijn communication networks", *IEEE Trans. Comput.* **C-34** (1985) 777–788.
7. J. Rothstein and A. Davis, "Bus automata, brains, and mental models", *IEEE Trans. Syst. Man Cybern.* **18** (1988) 522–531.
8. M. A. Sridhar, "On the connectivity of the de Bruijn graph", *Infor. Proc. Lett.* **27** (1988) 315–318.
9. S. P. Kartashev and S. I. Kartashev, "Analysis and synthesis of dynamic multicomputer networks that reconfigure into rings, trees, and stars", *IEEE Trans. Comput.* **C-36** (1987) 823–844.
10. C. L. Wu, T. Y. Feng, and M. C. Lin, "Star: A local network system for real-time management of imagery data", *IEEE Trans. Comput.* **C-31** (1982) 923–933.
11. L. Snyder, "Introduction to the configurable, highly parallel computer", *IEEE Comput.* **15** (1982) 47–56.
12. J. Beetem, M. Denneau, and D. Weingaarten, "The GF 11 supercomputer", *Proc. Ann. Symp. Comput. Arch.*, 1985, pp. 108–115.
13. D. K. Pradhan, "Dynamically restructurable fault-tolerant processor network architectures", *IEEE Trans. Comput.* **C-34** (1985) 434–447.
14. R. Miller, V. K. P. Kumar, D. Reisis, and Q. F. Stout, "Meshes with reconfigurable buses", *Proc. Fifth MIT Conf. Adv. Res. VLSI*, 1988, pp. 163–178.
15. R. Negrini, M. Sami, and R. Stefanelli, "Fault-tolerance techniques for array structures used in supercomputing" *IEEE Comput.* (1986) 78–87.

16. K. P. Belkhale and P. Banerjee, "Reconfiguration strategies for VLSI processor arrays and trees using a modified diogenes approach", *IEEE Trans. Comput.* **41** (1992) 83–96.
17. B. Monien and H. Sudburough, "Embedding one interconnection network in another", *Comput. Suppl.* **7** (1990) 257–282.
18. N. N. Biswas and S. Srinivas, "A reconfigurable tree architecture with multistage interconnection network", *IEEE Trans. Comput.* **39** (1990) 1481–1485.
19. D. H. Lawrie, "Access and alignment of data in an array processor", *IEEE Trans. Comput.* **C-24** (1975) 1145–1155.