Parallel Implementation of Divide-and-Conquer Algorithms on Binary de Bruijn Networks

Xiaoxiong Zhong, Sanjay Rajopadhye
Virginia M. Lo

CIS-TR-91-21
September 1991

Department of Computer and Information Science
University of Oregon

Parallel Implementation of Divide-and-Conquer Algorithms on Binary de Bruijn Networks [*]

Xiaoxiong Zhong, Sanjay Rajopadhye[†] Virginia M. Lo[‡]

CIS-TR-91-21
September 1991

Abstract

We study the problem of parallel implementation of divide-and-conquer algorithms on binary de Bruijn network. We model the divide-and-conquer algorithm as a temporal binomial tree computation structure which we show to be an efficient structure for this type of algorithms. We then discuss the problem of mapping such task structures (assigning tasks to processors and routing messages through communication channels) to a binary de Bruijn network of the same size. Two cases are considered where message volumes are of uniform and logarithmically decreasing (increasing) weights. A single mapping is proposed for both cases and the mapping has *average extra dilation* 1 and is *communication link contention-free.* A lower bound for the total extra dilation of any mapping from uniform-weighted binomial tree to an arbitrary degree-4 network (a binary de Bruijn network has degree 4) is also developed to show that the mapping is asymptotically optimal with respective to the average extra dilation. The mapping is well suited to a binary de Bruijn network with a wormhole or circuit switching communication scheme.

Department of Computer and Information Science
University of Oregon

[*]Supported by Oregon Advanced Computing Institue (OACIS)
[†]Partially supported by NSF-grant MIP-8802454
[‡]Partially supported by NSF-grant CCR-8808532

1 Introduction

Divide and Conquer method is an important parallel programming paradigm [Nel87, Col89]. The problem of implementation of such algorithms onto various parallel architectures such as hypercubes, tree machines, mesh connected computers, etc. has been well studied [Col89,LRG$^+$90,AS88]. As an interconnection network, a de Bruijn network has many superior features such as bounded degree, logarithmic diameter and high connectivity. Recently, there hasbeen increasing interests in exploring such networks as future underlying interconnection networks for multiprocessor systems [Sch74, SP89,PR82,RB90]. A prototype machine for Vitterbi decoding in the Galileo space problem is being built at Jet Propulsion Lab. This paper addresses the problem of implementing divide and conquer algorithms on such networks.

Traditionally, the task structure of a divide and conquer algorithm is modeled as a complete binary tree. It has been pointed out, however, that a binomial tree is a more efficient task graph than a complete binary tree [AS88,Col89,LRG$^+$90]. The mapping problem arises from the topological mismatch between the task graph and the underlying interconnection structure of the target machine (in our case, a binary de Bruijn network). To solve this problem, the graph embedding approach [Ros88,Bok87] can be used to embed the task graph to the target architecture. The standard metric used in this approach is to minimize the maximum dilation. This is well justified for a network with store and forward communication scheme. However, recent communication technology development has made a fast communication possible. New communication schemes such as wormhole [DS86],circuit switching or virtual cut through have been used in many commercial machines. In these machines, not only the distance a message has to be sent is important, but the message traffic congestion also plays an important role in the efficient implementation of the program. To reduce the message traffic, a good routing should be designed. This certainly increases the difficulty for the mapping problem. Fortunately, for divide and conquer type algorithms, there is a very regular *temporal* message traffic pattern. Such temporal information can be used to develop mappings which have light traffic congestion.

We consider two different types of message traffic for a divide and conquer algorithm in the paper. The first is the *uniform weighted case* (i.e., all messages are of the same volume) and the second is the *logarithmically weighted case* where message weights are logarithmically decreasing (or increasing) level by level. Many parallel algorithms have one or both such traffic patterns. Some well known examples include mergesort, finding max, parallel voting and broadcasting, multiplication of a sequence of matrices (the dividing mode of this algorithm is the logarithmically weighted case and the aggregating mode is the uniform weighted case) and parallel prefix. A single mapping to the binary de Bruijn network is proposed for both cases. For both cases, the *average extra dilation*[*] of the mapping is 1. For the message traffic congestion, the mapping is link contention-free, i.e., there is no communication conflict for any pair of simultaneous messages. For the uniform weight case, a lower bound of 1/16 for the average extra dilation of any mapping of a to an arbitrary degree-4 network (a binary de Bruijn network has degree 4) is developed. This indicates our mapping is asymptotically optimal.

The rest of the paper is organized as follows. In Section 2, we discuss the binomial tree as the task graph for a degree-2 divide and conquer algorithm and its properties. In Section 3, we present some useful properties for de Bruijn network. In Section 4, the evaluation metrics for the mapping are discussed. In Section 5, the mapping is proposed and evaluated. In Section 6, a bound for the total extra dilation is developed. Section 7 concludes the paper.

2 Divide and Conquer Algorithms and Binomial Trees

A classic task graph structure for a degree-2 divide and conquer algorithm is a complete binary tree where, the root first starts to divide the work into two approximately equal parts to its two children and once an interior node receives the work, it also divides it evenly to its children. After the leaves receive their work, they start to do compute and the results are aggregated level by level back to the root. Such approach, as it has been pointed out in [AS88,Col89,LRG$^+$90], is not efficient since after work has been divided and sent out, a node is idle until it receives the result. A more efficient way is to adopt "keep half, send half" policy as follows.

Step 1 Give the initial process the problem to be solved;

[*]the concept of *average extra dilation* will be defined later

Step 2 Let all processes divide their problem into two sub-problems, spawn a new child process and pass on *one of the subproblems* to it and *keep the other half to itself*;

Step 3 Repeat **Step 2** until the problems become sufficiently "small";

Step 4 All processes solve their problem locally.

Step 5 The results are passed up to parents and merged *in the reverse of the order in which the sub-problems were passed down the tree*

Step 6 Continue Step 5 recursively till the values reach the root.

Compared with the complete binary tree approach, for the same divide and conquer algorithm, the "keep half, send half" approach only needs approximately half number of processes (thus, it has twice efficiency than the complete binary tree approach) and internalizes almost half of messages. Such an approach has been used in [AS88,Col89]. We have shown that the above task structure corresponds to a binomial tree and studied it in more detail in [LRG+90]. In this paper, we use the binomial tree as the static task structure for a divide and conquer algorithm. A binomial tree $B(n)$ is defined inductively as follows.

Definition 1 A single node is a binomial tree $B(0)$. We construct $B(n)$ by attaching the root of one $B(n-1)$ to the root of another $B(n-1)$. Fig. 1 shows the construction.

$B(n)$ has 2^n number of nodes. The root r of $B(n)$ has n children which in turn are the roots of $B(n-1), B(n-2), \ldots, B(0)$ (see Fig. 1). We use the convention that the ith-child of r is the root of $B(n-i)$.

There exists a contraction Q from a complete binary tree $CBT(n+1)$ to a binomial tree $B(n)$ which is defined as follows, for any node u in $CBT(n+1)$, if u is a left child of its parent v, then u and v are contracted into the same node. Q can be performed by starting a leaf in $CBT(n+1)$ and then tracing through the left-child relation towards the root until a node which is not a right child of its parent or the root of $CBT(n+1)$ is reached. Fig. 2 shows this contraction.

The binomial tree approach has two major modes: **Step 2, 3** is a mode for problem dividing mode. **Step 5,6** is a mode for result aggregating. There are two common

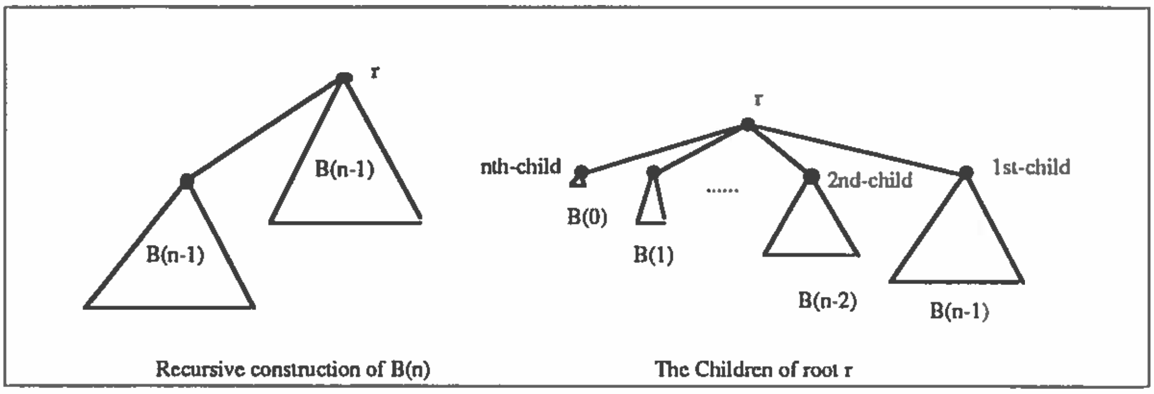

Figure 1: The Definition of a Binomial Tree

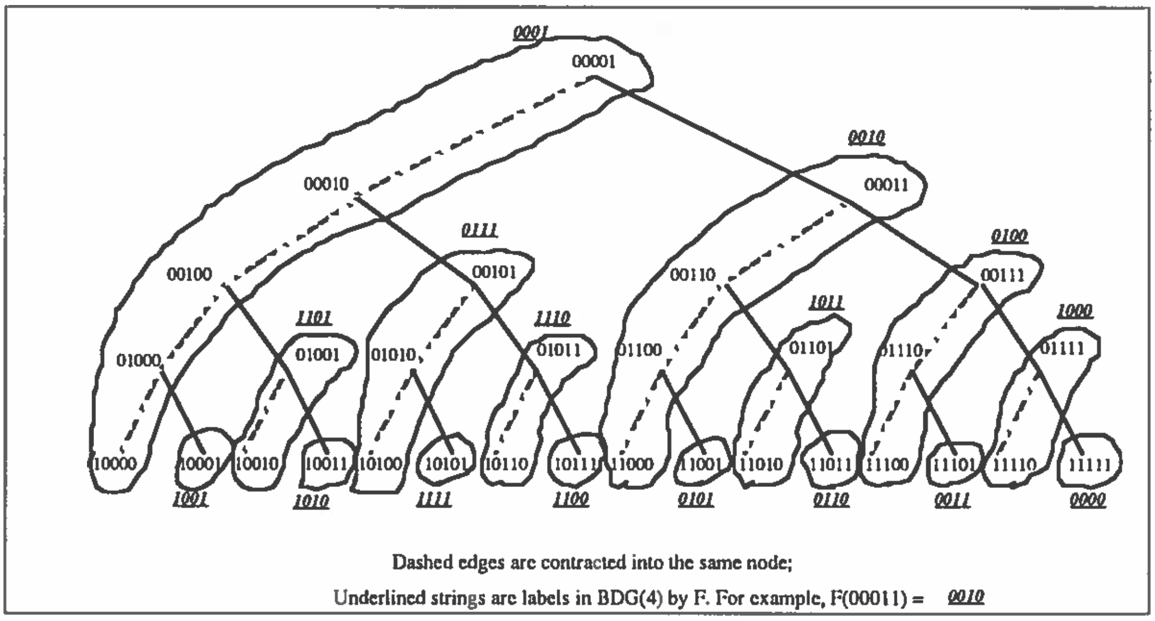
Dashed edges are contracted into the same node;
Underlined strings are labels in BDG(4) by F. For example, F(00011) = 0010

Figure 2: Contraction from $CBT(5)$ to $B(4)$ and the mapping from $B(4)$ to $BDG(4)$ by showing the function F

message traffic patterns in which we are interested in this paper. One is that the volumes of messages are the same. The other is that message volumes decrease (increases) by half at each iteration of **Step 2** (**Step 5** level by level in the dividing (aggregating) mode. We call the first *uniform weight case* and the second *logarithmically weighted* case. We note that the message traffic patterns in the dividing mode and the aggregating mode can be different. For example, to multiply a sequence of $n \times n$ matrices, in **Step 2**, a node divides the matrix sequence it has into two subsequences of the same length and sends one of them to its child. Therefore, in the dividing mode, weights are logarithmically decreasing. In the aggregating mode, however, a single result matrix is passed up to its parent and hence it is the uniform weight case.

Without loss of generality, we will only consider a single mode (dividing mode) for both cases. We also assume that the size of the original problem is 1.

In addition to the fact that the static task structure of the"keep half, send half" approach corresponds to a binomial tree, there is also a regular temporal message traffic pattern in the communication. For a binomial tree $B(n)$, in the dividing mode, there are a total of n communication phases where, in the ith-phase, all nodes in level i sends out messages to their first child, nodes in level $i-1$ sends to their second child,..., node in level 1 (the root) sends a message to its ith child. Furthermore, for the logarithmically weighted case,messages sent out in phase i are of the same volume $1/2^i$. Fig. 3 illustrates this temporal pattern.

The following lemma characterizes the message volume that a node in $B(n)$ can receive (a node only receives a message once in the whole dividing mode)

Lemma 1 Suppose u is a node in $B(n)$ with i children $(0 < i < n)$. It receives message with volume $1/2^{n-i}$ in the logarithmically decreasing case.

Proof: It is clear that all the leaves of $B(n)$ receive the same size of message, namely, $1/2^n$. It is also true that a node u with i children distributes the message of volume $M(u)$ that it receives $1/2$ to its 1st-child, $1/2^2$ to its 2nd-child,..., and $1/2^i$ to its ith-child, which is a leaf node in $B(n)$. Therefore, we have $\frac{1}{2^i}M(u) = \frac{1}{2^n}$, which implies that $M(u) = \frac{1}{2^{n-i}}$. ∎

Since a node only receives a message in the whole dividing mode, we can view the binomial task structure as a weighted graph where the weight of an edge is the volume of the message transmitted over this edge in the whole mode. In the uniform case, weights are all 1.

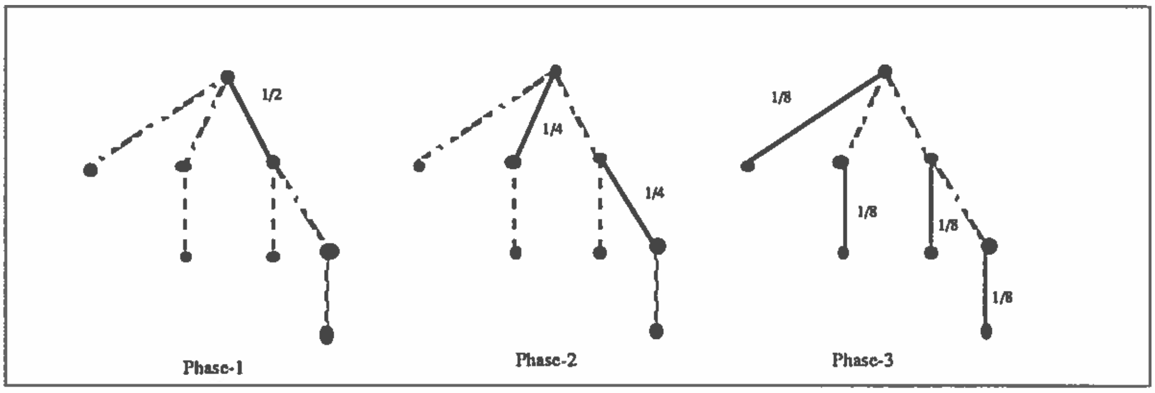

Figure 3: Three phases in $B(3)$ (colored binomial tree structure). Labels of edges are the message volumes. Dashed edges are inactive edges in that phase

3 The Binary de Bruijn Network

In this section, properties for a binary de Bruijn network are studied.

Definition 2 A de Bruijn network(graph) $DG(r,k)$ [SP89] has r^k nodes. Each node is represented as a k-digit number of r-radix: $a_{k-1}a_{k-2}\ldots a_0$ where $a_i \in \{0,\ldots,r-1\}$. Every node $a_{k-1}a_{k-2}\ldots a_0$ is connected to the nodes: $a_{k-2}\ldots a_0x$ (by left shifting) and $xa_{k-1}a_{k-2}\ldots a_1$ (by right shifting) where $x \in \{0,\ldots,r-1\}$. $DG(2,k)$ is called a binary de Bruijn network and denoted by $BDG(k)$.

Figure 4 shows the graph of $BDG(4)$.It is easy to see that there are a total of 2^k nodes in $BDG(k)$, the maximum degree of nodes in a $BDG(k)$ is 4 and the diameter is k.

Property 3.1 *[SP89] There are at least four distinct complete binary trees of depth k (denoted as $CBT(k)$) as subgraphs in a $BDG(k)$.*

Proof: We reproduce the proof in [SP89] because it will be useful later. Two of the CBTs are constructed as follows: To construct the $CBTs$, start at $\bar{x}^{k-1}x$ for $x \in \{0,1\}$ [*] as the root, left shift to $\bar{x}^{k-2}x0$ to construct the left child, and left shift to $\bar{x}^{k-2}x1$ to construct the right child. Inductively, any node $\bar{x}^m xz$ has the

[*]We use $\bar{x}$ to denote the complement of x, and x^m to denote the string with m repetitions of x

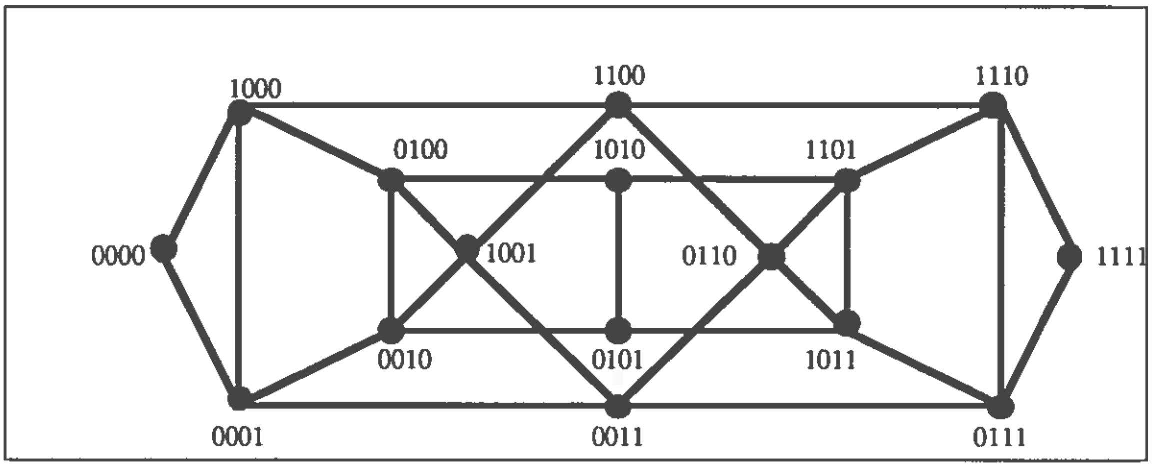

Figure 4: A DBG(4) and its labeling

left child labeled $\bar{x}^{m-1}xz0$ and the right child $\bar{x}^{m-1}xz1$. This procedure can be repeated until we reach a string whose leftmost bit is x, which is the leaf of the CBT to be constructed. In the similar way, we can construct two more distinct $CBTs$ by starting at $x\bar{x}^{k-1}$ as the root, and using right shifting instead of left shifting. ∎

Figure 5 shows the $CBT(5)$ whose root is 00001 in $BDG(5)$.

We now reveal an important contraction existing between $BDG(n)$ and $BDG(n-1)$. First, let us define a function F which maps a binary string of length n to a binary string of length $n-1$ as follows: $F(x_1x_2\ldots x_{n-1}x_n) = \hat{x}_1\hat{x}_2\ldots\hat{x}_{n-1}$ where $\hat{x}_i = x_i \oplus x_{i+1}$ for $i < n$ where $\oplus$ is the exclusive-or operation.

Lemma 2 F is a two-to-one function which maps both $x_1x_2\ldots x_{n-1}x_n$ and its binary complement string $\bar{x}_1\bar{x}_2\ldots\bar{x}_{n-1}\bar{x}_n$ to the same string.

Proof: It is easy to see that F maps both $x_1x_2\ \ldots\ x_{n-1}x_n$ and $\bar{x}_1\bar{x}_2\ \ldots\ \bar{x}_{n-1}\bar{x}_n$ to the same string $\hat{x}_1\hat{x}_2\ldots\hat{x}_{n-1}$ where $\hat{x}_i = x_i \oplus x_{i+1}$ for $i < n$ because $x \oplus y = \bar{x} \oplus \bar{y}$. Conversely, the strings $x_1x_2\ldots x_{n-1}x_n$ which are mapped to $y_1y_2\ldots y_{n-1}$ are the solutions to the system of equations $x_i \oplus x_{i+1} = y_i$ for $i \leq (n-1)$. Since $x \oplus y = z$ iff $y = x \oplus z$ for any binary x, y, z, we have the following equivalent system of equations.

$$x_2 \quad = \quad y_1 \oplus x_1$$

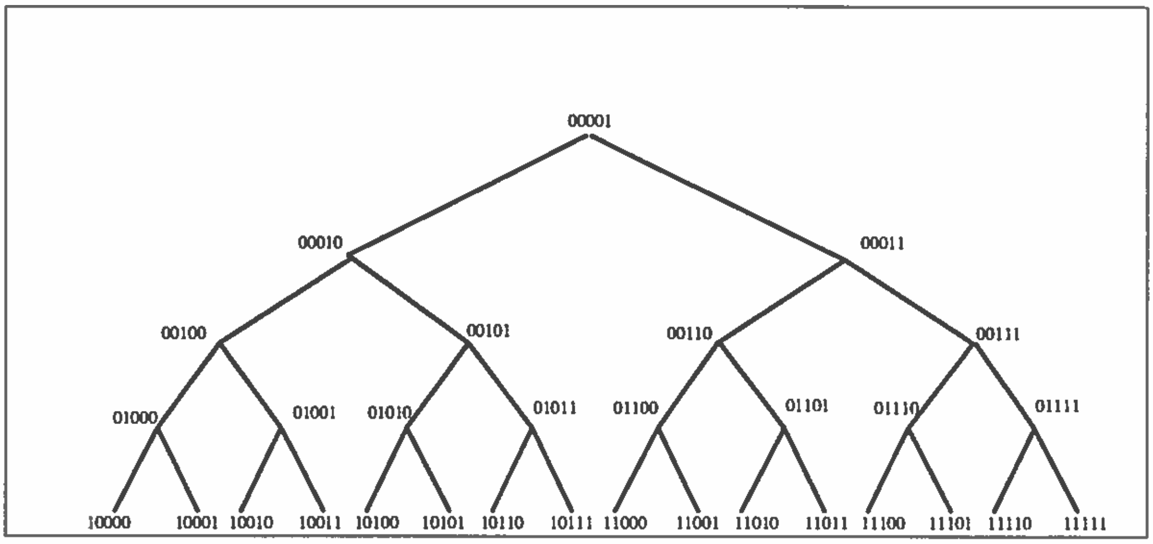

Figure 5: A CBT(5) and its labeling

$$\begin{aligned} x_3 &= y_2 \oplus y_1 \oplus x_1 \\ &\dots \\ x_n &= y_{n-1} \oplus y_{n-2} \oplus \dots \oplus y_1 \oplus x_1 \end{aligned}$$

Thus, $x_1 = 0$ and $x_1 = 1$ yield the only two distinct solutions. ▮

The range of F is thus all the 2^{n-1} distinct binary strings of length $n-1$. If we view F as a contraction function for $BDG(n)$, the following property tells us that the contracted graph is exactly $BDG(n-1)$.

Property 3.2 $F(BDG(n)) = BDG(n-1)$ *(with nodes labeled in the same way).*

Proof: Assume the contracted graph is G. First, we show that any edge in G is an edge in $BDG(n-1)$. There are two types of edges in $BDG(n)$, namely, those corresponding to left shifting and those corresponding to right shifting. A left shifting edge between nodes $x_1x_2\dots x_{n-1}x_n$ and $x_2x_3\dots x_nz$ yields the edge in G between the nodes $\hat{x}_1\hat{x}_2\dots\hat{x}_{n-1}$ and $\hat{x}_2\hat{x}_3\dots\hat{x}_{n-1}z'$ where $\hat{x}_i = x_i \oplus x_{i+1}$ for

$i < n$ and $z' = x_n \oplus z$, which is a left shifting edge between these two nodes in $BDG(n-1)$. A similar argument holds for right shifting edges.

Conversely, for any left shifting edge between $Y_1 = y_1 y_2 \dots y_{n-1}$ and $Y_2 = y_2 y_3 \dots y_{n-1} y$ in $BDG(n-1)$, let $X_1 = x_1 x_2 \dots x_{n-1} x_n$ be one of the nodes in $BDG(n)$ such that $F(X_1) = Y_1$, i.e., $y_i = x_i \oplus x_{i+1}$ for $i \le (n-1)$. Construct another node in $BDG(n)$ as $X_2 = x_2 x_3 \dots x_n x$ where $x = x_n \oplus y$. $F(X_2) = Y_2$ since $x_n \oplus x = x_n \oplus x_n \oplus y = y$. Because there is a left shifting edge between X_1 and X_2 in $BDG(n)$, this edge yields a left shifting edge between Y_1 and Y_2 in G. A similar argument holds for right shifting edges *. ∎

Therefore, the image of a path P in $BDG(n+1)$ under F is still a path in $BDG(n)$.

4 The Metrics

To solve the problem of the topological mismatch between the binomial tree and the binary de Bruijn network, two decisions should be made: how the nodes (tasks) of the binomial tree will be allocated to the nodes (processors) of the binary de Bruijn network and how the communication edges of the binomial tree will be laid out along the links of the processors. In this paper, we assume that the number of available processors is equal to the number of tasks so that each task can be assigned to a *unique* processor. This is reasonable since if the number of the tasks is greater than that of available processors, one can stop spawning tasks earlier.

Formally, we want to find a mapping which is specified by two functions *map-node* and *map-edge* for a given binomial tree $B(n) = (V_B, E_B)$ and a binary de Bruijn network $BDG(n) = (V_D, E_D)$ such that *map-node* maps nodes in $B(n)$ to nodes in $BDG(n)$ and *map-edge* maps an edge $(\langle a, b \rangle)$ in $B(n)$ to a path from *map-node*(a) to *map-node*(b) in $BDG(n)$. *map-edge* routes messages from a task to another task through a route in the $BDG(n)$. To unify the discussions for both uniform and logarithmically decreasing cases, we associate $B(n)$ with an edge weight function W_B.

To efficiently implement a parallel algorithm, we should minimize the cost of interprocessor communication (IPC). The overhead of IPC can be minimized by mapping

*The above contraction property between two de Bruijn graphs is also independently discovered by Bose [Bos].

tasks that communicate as close to each other as possible and by avoiding traffic congestion on the communication channels of of the interconnection network by careful routing. The following are the commonly used metrics.

Definition 3 The **dilation** of an edge $e \in E_B$ is

$$W_B(e) \times |P|$$

where *map-edge*$(e) = P$ and $|P|$ is the length of path P. The **extra dilation** of e is defined as its dilation minus $W_B(e)$.

Ideally the dilation of every edge e in E_C should be $W_B(e)$ (for uniform case, 1), but this may be impossible to be achieved in many cases. Two metrics can be used. One is *maximum dilation* and the other *average dilation.*

Definition 4 The **maximum dilation** of a mapping is $max\{d{:}d{=}dilation\ of\ e,\ e \in E_B\}$.

Maximum dilation is the common standard metric used in the embedding literature [Ros88] where an important goal is to study how one interconnection network can emulate another network [Ull84]. However,this metric is more useful for a uniform mapping than a weighted mapping. It is also limited since it does not measure a mapping by the dilation of all edges but by only the worst case. A more appropriate metric is the *average dilation* which is defined as follows.

Definition 5 The **average dilation** of a mapping is

$$\frac{\sum_{e \in E_B} dilation\ of\ e}{\sum_{e \in E_B} W_B(e)}.$$

The **average extra dilation** is defined similarly by replacing dilation of e with extra dilation of e in the above formula. It is easy to see that the **average extra dilation** is the **average dilation** minus 1.

The average dilation is at best 1.

The above metrics only characterize the topological aspect of a mapping. Recent developments in network communication techniques has made the communication traffic be a dominating factor in an execution of a parallel program. In the current communication shcemes such as wormhole, circuit switching or virtual cut through, the communication conflict is much more important than the distance a message has to be sent. For example, in a wormhole routing, a message is divided into small pieces called flits which are routed through the network in a pipeline fashion. The distance is negligible if a message is big enough. However, in such routing shceme, once a message is blocked due to the contention for a communication channel, the whole message stays in the network, occupying the channels (which may in turn block other messages) until the blocked channel is released. This introduces an important concept called **link contention.** Intuitively, a mapping is called **link contention free** if there is no conflict for any pair of simultaneous messages. The traditional way to find a link contention-free mapping is to ensure that message paths are pairwise edge disjoint. However,this does not explore the temporal information. For a binomial tree divide and conquer algorithm, as we mentioned before, messages are sent out at different phases in a regular pattern. Therefore, for such a particular class of algorithms, a mapping is link contention free as long as messages sent in the same communication phase are routed through edge disjoint paths. We have developed such mappings of binomial trees to hypercubes and meshes in [LRG+90]. The mapping presented in the next section also has such property.

5 The Mapping

We first present a mapping D from the binomial tree $B(n)$ to a binary de Bruijn network $BDG(n)$, and then evaluate its metrics for both uniform and logarithmically decreasing weight cases.

5.1 The Construction of the Mapping

Mapping D can be intuitively described as follows. First, we pick a complete binary tree CBT in $BDG(n+1)$. For this, we choose the one whose root is 0^n1 and denote it as $CBT(n+1)$. We then perform contraction Q on $CBT(n+1)$. In this contraction, every cluster contains a single node whose label is odd (the "topmost" node in the cluster). Thus, every odd node in $CBT(n+1)$ can uniquely identify a node in the binomial tree $B(n)$. Also, edges in $B(n)$ are mapped to paths in this $BDG(n+1)$ which are the paths in $CBT(n+1)$. Since the $CBT(n+1)$ is in the de Bruijn graph $BDG(n+1)$, we can map nodes with odd labels in the $BDG(n+1)$ to the smaller $BDG(n)$ by contraction F, which in turn gives us a mapping from the $B(n)$ to the $BDG(n)$. Since paths are preserved under F, those paths in $BDG(n+1)$ corresponding to edges in $B(n)$ are preserved in $BDG(n)$, which gives *map-edge* of D. This mapping is formalized in the following. First, we note which nodes in $CBT(n+1)$ are contracted by Q to the same node.

Lemma 3 Nodes labeled $0^m z1, 0^{m-1}z10, \ldots, z10^m$, where z is a binary string of length $n-m$ whose leftmost bit is 1 if $n > m$ and is the empty string for any $n = m$ are contracted into the same node by Q (see Fig. 6).

Proof: From the construction of $CBT(n+1)$, it is easy to see that the label of any leaf in $CBT(n+1)$ can be always factorized as $z10^m$ where z is either the empty string (if $m = n$) or is a string of length $n-m$ whose leftmost bit is 1 (note,the leftmost bit of the label of a leaf is always 1). Since the leaf $z10^m$ is the left child of $0z10^{m-1}$, which in turn is the left child of 0^2z10^{m-2}, which in turn is ..., and $0^m z1$ is not the left child of its parent, based on contraction Q, we know nodes $0^m z1, 0^{m-1}z10, \ldots, z10^m$ are contracted into a single node by Q. Fig. 6 shows this contraction. ∎

Thus, a node in $B(n)$ is represented by a cluster of nodes $\{0^m z1, 0^{m-1}z10, \ldots, z10^m\}$. Furthermore, we notice this cluster of nodes can be uniquely determined by the first node labeled $0^m z1$ (the only odd string label among these labels). Hence, we can label a node in $B(n)$ by $0^m z1$ where z is either the empty string (if $m = n$) or is a string of length $n-m$ whose leftmost bit is 1.

We are now ready to describe the mapping D.

Definition 6 The mapping D is defined by the two functions:

$$\begin{aligned} map-node(0^m z1) &= F(0^m z1) \\ map-edge(< 0^m z1, 0^{m-i}z10^{i-1}1 >) &= path\{F(0^m z1), F(0^{m-1}z10), \ldots, \\ &\quad\ F(0^{m-i+1}z10^{i-1}), F(0^{m-i}z10^{i-1}1) \end{aligned}$$

Fig. 6 shows the *map-edge* of D. Fig. 2 shows the mapping from $B(4)$ to $BDG(4)$. We need to formally prove that the above definition is well-defined.

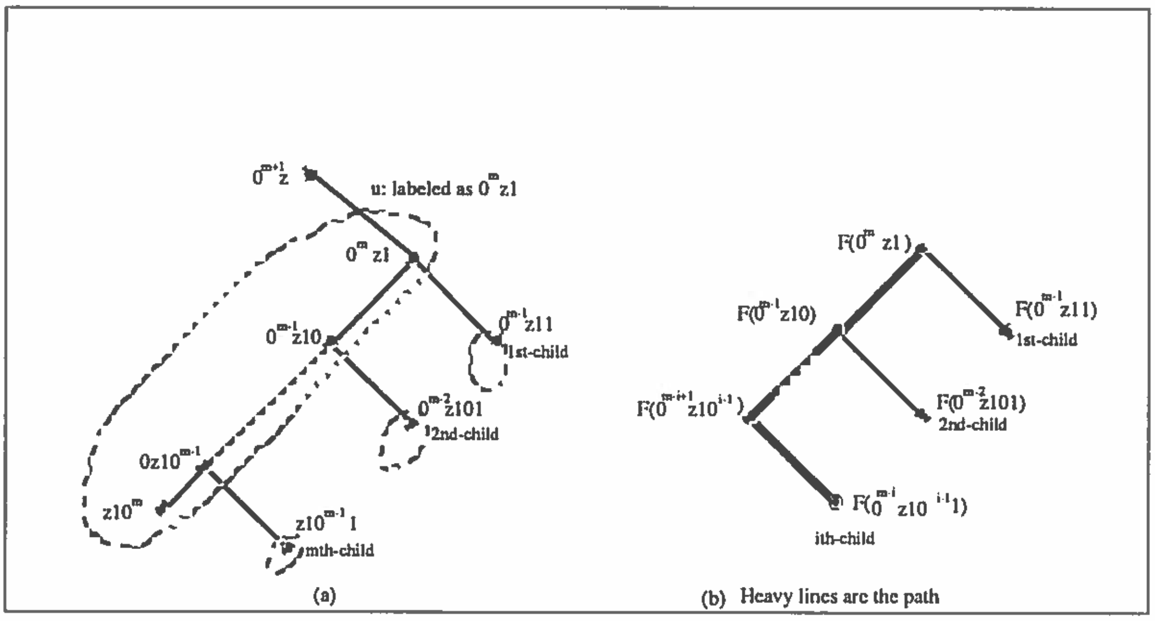

Figure 6: (a). u has m children (note $0^m z1$ is the right child of its parent). (b). the edge between $0^m z1$ and $0^{m-i}z10^{i-1}1$ in $B(n)$ is mapped to a path in $BDG(n)$

Property 5.1 *D is a mapping from $B(n)$ to $BDG(n)$.*

Proof: There are totally 2^n nodes labeled as $0^m z1$ in $CBT(n+1)$ (in fact, all the interior nodes and leaves which are right children). All of these nodes will be mapped to distinct nodes in $BDG(n)$ under F since, based on Lemma 2, the other node which is contracted by F to the same node as $0^m z1$ is its complement. Since $0^m z1$ ends with 1 (i.e., odd string), the label of its complement must end with 0 (i.e, even string). The label of its complement can not be $0^m z1$. Therefore, *map-node* of D is a bijective function. Furthermore, since $0^m z1, 0^{m-1}z10, \ldots, 0^{m-i+1}z10^{i-1}, 0^{m-i}z10^{i-1}1$ is a path in $CBT(n+1)$, it is also a path in $BDG(n+1)$. Since paths in $BDG(n+1)$ are preserved under F, we know $F(0^m z1)$, $F(0^{m-1}z10)$, ...,$F(0^{m-i+1}z10^{i-1})$, $F(0^{m-i}z10^{i-1}1)$ is indeed a path in $BDG(n)$. ∎

5.2 Evaluation of the mapping

1. Uniform Case:

Property 5.2 *The average dilation of D is 2 (asymptotically).*

Proof: First of all, notice for any $0^m z1$ where z satisfies the restriction that its leftmost bit is 1, the children of the node labeled as $0^m z1$ in $B(n)$ are mapped to nodes $F(0^{m-k}z10^{k-1}1)$ for $1 \leq k \leq m$ in $BDG(n)$. The path between $0^m z1$ and $0^{m-k}z10^{k-1}1$ in $CBT(n)$ (which is in $BDG(n+1)$ too) is of length k (see Fig. 6). This path is preserved under the contraction F in $BDG(n)$. Therefore, the dilation between $F(0^m z1)$ and $F(0^{m-k}z10^{k-1}1)$ is at most k. Because there is a node labeled as $0^m z1$ for $m = n$ (i.e. z is the empty string) and there are 2^{n-m-1} nodes labeled as $0^m z1$ for $0 \leq m \leq (n-1)$, we have

$$TotalDilation = \sum_{i=1}^{n} i + \sum_{m=0}^{n-1}\sum_{k=0}^{m} k2^{n-m-1}$$

Evaluating the above summation gives us the $TotalDilation = 2^{n+1} + \Theta(n^2)$. Therefore, the average dilation is asymptotically 2. ∎

Property 5.3 *The maximum dilation of D in the uniform case is at most n.*

Proof: The maximum dilation in the mapping of $B(n)$ to $BDG(n)$ is between $F(0^n 1)$ and $F(10^{n-1}1)$ under the path $F(0^n 1), F(0^{n-1}10), ..., F(10^{n-1}1)$. ∎

Property 5.4 *D is link contention-free.*

Proof: The edge between a node $0^m z1$ and its kth-child $0^{m-k}z10^{k-1}1$ ($1 \leq k \leq m$) in $B(n)$ is mapped by D to path $F(0^m z1), F(0^{m-1}z10), \ldots, F(0^{m-k+1}z10^{k-1})$, $F(0^{m-k}z10^{k-1}1)$ in $BDG(n)$ (which is the image of path P: $0^m z1, 0^{m-1}z10, \ldots$, $0^{m-k+1}z10^{k-1}$, $0^{m-k}z10^{k-1}1$ in $CBT(n+1)$). Now, consider any other edge between node $0^q y1$ and its its lth-child $0^{q-l}z10^{l-1}1$ in $B(n)$, there is also a corresponding path P' from $0^q y1$ to $0^{q-l}z10^{l-1}1$ in $CBT(n+1)$. It is easy to see that the internal nodes of path P are different from the internal nodes of path P' if $m \neq q$ or $z \neq y$. Furthermore, since the labels of all the internal nodes of any such path P in $CBT(n+1)$ end with 0 (i.e., even string), two distinct internal nodes are mapped by F to distinct nodes in the $BDG(n)$ (since they are not complemented with each other). Therefore, edges of $F(P)$ and $F(P')$ are disjoint if $F(0^m z1)$ is different from $F(0^q y1)$.

The only remaining case is for two paths between the same node $F(0^m z1)$ and two of its children (if any) in the $BDG(n)$. Since message passing along these two paths will happen in different phases, there will be no conflict too. This completes our proof. ∎

2. Logarithmically Weighted Case:

Observe that in the dividing mode of a binomial tree communication structure, a node u, after it has received its message (volume $M(u)$), sends out a message with volume $M(u)/2^i$ to its i-th child in the next i-th phase. Thus, the smaller the i is, the bigger the message volume it receives. This is also true in the conquering phase. To minimize the total weighted dilation between u and its children, we should therefore try to minimize the length of paths between u and its smaller i-th children.

Mapping D has such property. A node and its i-th child in $B(n)$ has a path of length i in the target binary de Bruijn graph $BDG(n)$. In the following, we prove that the average dilation of D is approximately 2.

Based on Lemma 1, in the binomial tree model $B(n)$, the message a node u labeled $0^m z1$ receives is $\frac{1}{2^{n-m}}$. This is because $0^m z1$ has m children (see Fig. 6). Therefore, the total dilation between u and its m children can be evaluated in the following way. The edge between u and its j-th child is mapped to path $F(0^m z1)$, $F(0^{m-1}z10)$, $\ldots$,$F(0^{m-i+1}z10^{i-1})$, $F(0^{m-i}z10^{i-1}1)$ in $BDG(n)$, whose length is j. The message volume u sends to its j-th child is $M(u)\frac{1}{2^j}$. Hence the total dilation between u and all its children in the dividing mode is

$$\frac{1}{2^{n-m}}\sum_{j=1}^{m} j\frac{1}{2^j} = \frac{1}{2^{n-m}}(2 - \frac{m+2}{2^m})$$

Since there are one node labeled 0^n1 and 2^{n-m-1} number of nodes labeled $0^m z1$ for $m < n$, the total dilation is

$$\sum_{j=1}^{n}\frac{j}{2^j} + \sum_{m=1}^{n-1} 2^{n-m-1}\frac{1}{2^{n-m}}(2 - \frac{m+2}{2^m}) = n - 1 + \frac{1}{2^n}$$

Property 5.5 *The average dilation of D for the logarithmically decreasing case is 2 (asymptotically).*

Proof: There are a total of n phases in the dividing mode of a binomial tree model $B(n)$. In each phase, the total message volume transmitted is 1/2. Therefore, based on the above discussion, we know that $\sum_{e\in E_C} W_B(e) = \frac{1}{2}n$. But the total dilation of D is $(n-1) + \frac{1}{2^n}$. Therefore, the average dilation is asymptotically 2. ∎

Property 5.6 *The maximum dilation of D in the logarithmically weighted case is at most $n/2^n$.*

Proof: Based on Lemma 1,a node u with i children receives a message with volume $1/2^{n-i}$. The message u sends to its jth child is of volume $1/2^{n-i+j}$. The path is of length j. Thus, the dilation is $j/2^{n-i+j}$. When $i = n$ and $j = n$, it achieves maximum $n/2^n$. ∎

A similar argument can be used to show that D is also contention free for the weighted case.

6 A lower bound for mapping to a degree-4 network

This section derives a lower bound for the total extra dilation of any mapping of a binomial tree to any degree-4 network A (which includes a binary de Bruijn network and a mesh). We prove that for the uniform mapping of $B(n)$ to a degree-4 network A, the total extra dilation must be at least $\Omega(\frac{1}{16}2^n)$, which means that the average extra

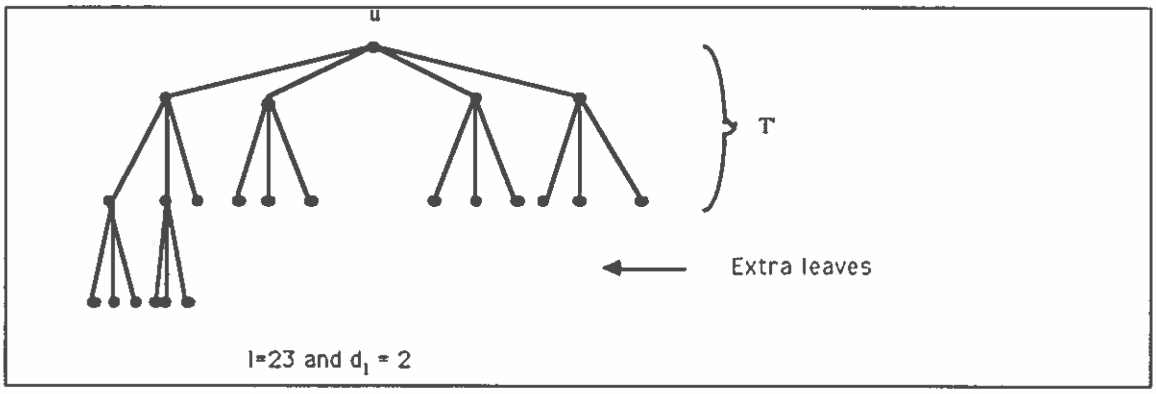

Figure 7: The best neighborhood tree structure for a mapping of $B(n)$ to a degree 4 network

dilation must be at least $\frac{1}{16}$. This indicates that mapping D, which has the average extra dilation 1, is satisfactory.

We first notice that in a $B(n)$, there are 2 nodes with degree n and there are 2^i nodes with degree $n-i$ for $1 \leq i \leq n-1$. We now consider for a node u with degree l, the least possible dilation between u and its $l-1$ children. Suppose that u is mapped to a node v in A, the best mapping we can achieve to minimize the dilation between u and its children is to map 4 of u's children to the four nodes which are adjacent to v, say v_1, v_2, v_3, v_4, and then embed 3 of u's children onto the three nodes adjacent to v_1 (recall A is a degree-4 network), and then to those adjacent to $v_2, v_3, \ldots$, until we map all of u's children. This results in a tree structure T in Fig. 7. We study the total extra dilation of T.

The structure of T is similar to that of a 3-heap tree except that there is one node-the root which has degree 4. Tree T can be divided into two parts: the first is the maximal complete tree T' where, every level is filled with nodes. The rest is the leaves which are not in T'(Fig. 7). Let the height of T' be d_l, we know there are $1 + 4 * 3^0 + 4 * 3 + \ldots + 4 * 3^{d_l - 1} = 2 * 3^{d_l} - 1$ number of nodes in T'. It is easy to see that $d_l = \lfloor \log_3 \frac{l+1}{2} \rfloor$ and the number of extra leaves in T is $l + 1 - 2 * 3^{d_l}$. For nodes in T', the total extra dilation is

$$0 + 1 * 4 * 3 + 2 * 4 * 3^2 + \ldots + (d_l - 1) * 4 * 3^{(d_l - 1)} = 4 \sum_{i=1}^{d_l - 1} i3^i = (2d_l - 3)3^{d_l} + 3$$

There are $(l + 1 - 2 * 3^{d_l})$ number of extra leaves (see Fig. 7). Each such node has extra dilation d_l, hence the total extra dilation for all such nodes is $d_l(l + 1 - 2 * 3^{d_l})$. Therefore, the total extra total dilation for u and its all children is $d_l(l+1) - 3^{d_l+1} + 3$. Notice that the above formula is only applicable for $l \geq 5$. Since there are 2^i number of nodes of degree $n - i$ in $B(n)$, the lower bound of the total extra dilation for all the nodes of $B(n)$ is

$$\sum_{i=1}^{n-5} 2^i[d_{n-i}(n - i + 1) - 3^{d_{n-i}+1} + 3]$$

Where $d_{n-i} = \lfloor \log_3 \frac{n-i+1}{2} \rfloor$. We consider all the terms such that $d_{n-i} = 1$, which dominates the value of the above summation. It is easy to see that when $n - 16 \leq i \leq n - 5$, $d_{n-i} = 1$ and we have

$$\sum_{i=n-16}^{n-5} 2^i[d_{n-i}(n - i + 1) - 3^{d_{n-i}+1} + 3]$$

$$= \sum_{i=n-16}^{n-5} 2^i[(n - i + 1) - 3^2 + 3] = 2^{n-4} - 13 * 2^{n-16}.$$

Therefore, we can see that the average extra dilation is at least

$$\frac{2^{n-4} - 13 * 2^{n-16}}{2^n - 1}$$

which is asymptotically $\frac{1}{16}$. Hence, we can conclude that the lower bound of the average extra dilation to map a $B(n)$ to a degree-4 network is at least $\frac{1}{16}$.

It can be noted that from the above argument, we can also derive an lower bound for the maximum dilation which, for an N node degree-4 network, the maximum dilation is at least $\log_3 log(N)$. But this does not imply that the mapping is not optimal with respect to the maximum dilation. The study of the optimality of the maximum dilation for a mapping from a binomial tree to a same size binary de Bruijn network remains open.

It should be noted that the above derivation can be generalized for any bounded degree network. For any mapping of a $B(n)$ to any degree-d network, we can similarly prove that the lower bound of the total extra dilation is $c2^n$ for some constant c. Therefore, we consider a mapping whose average extra dilation is a small constant to be a good mapping.

7 Conclusions

In this paper, we have presented a mapping to implement degree-2 divide and conquer algorithms on a binary de Bruijn network. For both uniform and logarithmically weighted cases, the mapping has average extra dilation 1, which is proved asymptotically optimal. The mapping also has an important property, link contention free. This means that the mapping is well suited to a de Bruijn network which has a new communication routing scheme such as wormhole,circuit switching or virtual cut through.

References

[AS88] W.C. Athas and C.L. Seitz. Multicomputers:Message-passing Concurrent Computers. *IEEE Computer*, pages 9–23, August 1988.

[Bok87] S.H. Bokhari. *Assignment problems in parallel and distributed computing.* Kluwer Academic Publishers, 1987.

[Bos] Bella Bose. Personal communication.

[Col89] M.I. Cole. *Algorithmic Skeletons:Structures Management of Parallel Computation.* MIT Press, 1989. Research Monographs in Parallel and Distributed Computing.

[DS86] W.J. Dally and C.L. Seitz. The torus routing chip. *Distributed Computing*, 1(3):187–196, October 1986.

[LRG+90] V.M. Lo, S. Rajopadhye, S. Gupta, D. Kelsen, M.A. Mohamed, and J. Telle. Mapping divide-and-conquer algorithms to parallel architectures. In *Proceed-*

ings of International Conference on Parallel Processing, pages III:128–135, 1990.

[Nel87] P.A. Nelson. *Parallel Programming Paradigms.* PhD thesis, Computer Science Department, University of Washington, July 1987.

[PR82] D.K. Pradhan and S.M. Reddy. A fault-tolerant communication architecture for distributed systems. *IEEE Transactions on Computers*, C-31(9):863–870, Sept. 1982.

[RB90] R. Rowley and B. Bose. On Necklaces in Shuffle-Exchange and de Bruijn Networks. In *Proceedings of International Conference on Parallel Processing*, pages I:347–350, August 1990.

[Ros88] A.L. Rosenberg. Graph embedding 1988: recent breakthroughs new directions. Technical Report 88-28, University of Massachusetts at Amherst, March 1988.

[Sch74] M.L. Schlumberger. *De Bruijn Communications Networks.* PhD thesis, Dept. of Computer Scienc, Stanford University, June 1974.

[SP89] M.R. Samatham and D.K. Pradhan. The de Bruijn multiprocessor network: a versatile parallel processing and sorting network for VLSI. *IEEE Transactions on Computers*, 38(4):567–581, April 1989.

[Ull84] J..D. Ullman. *Computational aspects of VLSI.* Computer Sciencce Press, Rockville,MD, 1984.