Lower Bounds on the Loading of Degree-2 Multiple Bus Networks for Binary-Tree Algorithms

Hettihe P. Dharmasena Ramachandran Vaidyanathan*
Department of Electrical & Computer Engineering
Louisiana State University, Baton Rouge, LA 70803-5901, USA
{evdhar,vaidy}@ee.lsu.edu

Abstract

A binary-tree algorithm, Bin(n), proceeds level-by-level from the leaves of a 2^n-leaf balanced binary tree to its root. This paper deals with running binary-tree algorithms on multiple bus networks (MBNs) in which processors communicate via buses. Every "binary-tree MBN" has a degree (maximum number of buses connected to a processor) of at least 2. There exists a degree-2 MBN [15] for Bin(n) that has a loading (maximum number of processors connected to a bus) of $\Theta(n)$. For any MBN that runs Bin(n) optimally, the loading was recently proved to be $\Omega(n^{\frac{1}{2}})$ [3]. In this paper, we narrow the gap between the results in [3, 15] by deriving a tighter lower bound of $\Omega(n^{\frac{2}{3}})$. We also establish a tradeoff between the speed and loading of degree-2 binary-tree MBNs.

1. Introduction

A Multiple Bus Network (MBN) consists of a set of processors connected via a set of buses. A pair of processors connected to a common bus may communicate with each other in one unit of time; only one piece of information may use a bus at any given point in time, however.

MBNs have several advantages over traditional point-to-point networks, primarily due to the fact that a bus, being a shared resource, can be used more flexibly and efficiently than a dedicated link. They also capture the essential features of systems with shared resources.

Previously, Kulasinghe and El-Amawy have presented a methodology for implementing vertex-symmetric point-to-point networks on MBNs [10] and have shown that in general, obtaining such an implementation with minimal cost is an NP-Hard problem [9]. Dighe *et al.* [4] have proposed an MBN alternative to trees, rings, meshes, and meshes of trees. Kamath and Vaidyanathan [8] have studied the problem of running weak hypercube algorithms on MBNs. The broadcast communication model (BCM) [13] is another embodiment of an MBN. Buses have also been used in a variety of ways to enhance the mesh topology [1, 2, 5, 6, 14] and in reconfigurable systems [7, 12].

Binary-tree algorithms [11] proceed level-by-level from the leaves to the root of a balanced binary tree. They have applications in a large number of problems (such as maximum, parity, prefix computation and polynomial evaluation) that use semigroup operations. Moreover, since they require an important interconnection structure (namely a binary tree), any network suitable for binary-tree algorithms is likely to be suitable for other applications as well. We will denote a binary-tree algorithm with $N = 2^n$ inputs by $Bin(n)$. Although considerable work on semigroup operations has been reported for enhanced meshes (for example [14]), they are for most part restricted by architectural features that only allow polynomial time solutions. Except in Section 4, we will only consider "optimal MBNs" that run $Bin(n)$ optimally in n steps.

This paper deals with the relationship between three important parameters of "binary-tree MBNs": (1) Speed, (2) Degree (maximum number of buses that may be connected to a processor), and (3) Loading (maximum number of processors that may be connected to a bus). For $n \geq 2$, an optimal MBN for $Bin(n)$ has a degree of at least 2 [3]. A degree-2, $\Theta(n)$-loading, optimal MBN for $Bin(n)$ has been proposed [15]. For any degree-2 MBN running $Bin(n)$ optimally, the loading was recently proved by the authors to be $\Omega(n^{\frac{1}{2}})$ [3]. In this paper, we narrow the gap between the results in [3, 15] by establishing a tighter lower

*Supported in part by the National Science Foundation under grant number CCR-9503882.

bound of $\Omega(n^{\frac{2}{3}})$.

We also establish a tradeoff between the speed and loading of degree-2 binary-tree MBNs. We show that a degree-2 MBN can have constant loading, if it is permitted to run the binary-tree algorithm more slowly. In particular, we show that a degree-2, MBN can have a constant loading, if and only if it uses $\Omega(n)$ steps beyond the optimal needed to run $Bin(n)$.

In the next section we present some preliminary ideas and results for MBNs. Section 3 is devoted to the general lower bound. In Section 4 we derive the relationship between the speed and loading of degree-2 binary-tree MBNs. In Section 5 we summarize our results and make some concluding remarks.

2. Preliminaries

Multiple Bus Networks: A $P \times B$ *Multiple Bus Network* (*MBN*) has P processors and B buses. Each processor is connected to a subset of the set of buses. The processors are assumed to work synchronously (SIMD model). Two processors may communicate in one step, provided they are connected to a common bus. However, only one piece of information may use a bus at any given point in time.

The number of buses to which a processor is connected is called the *degree of the processor*. The largest of the degrees of all processors is called the *degree of the MBN*. The number of processors connected to a bus is called the *loading of the bus*. The largest of the loadings of all buses is called the *loading of the MBN*. The degree and loading are important parameters that determine the cost and implementability of the MBN. The degree of an MBN is indicative of the number of input/output ports needed per processor. A large loading can introduce a significant delay or attenuation of the signal traversing a bus.

Figure 1(a) shows an 8×4 MBN whose degree and loading are 2 and 4, respectively.

Binary-Tree Algorithms: A *binary-tree algorithm*, $Bin(n)$, reduces 2^n inputs to a single result, for integer $n \geq 1$. The algorithm can be viewed as a 2^n-leaf, balanced binary tree, $\mathcal{T}(n)$, whose leaves represent the inputs and the root represents the final result; the internal nodes of $\mathcal{T}(n)$ represent partial results. Figure 1(b) shows $\mathcal{T}(3)$; the labels of the nodes and edges are explained later. The algorithm proceeds level-by-level, applying at each internal node v, a binary operation to the inputs/partial results of the two children of v.

We note that $Bin(n)$ describes a class of algorithms, rather than a class of problems (or reduction operations) that can be implemented as a binary tree. Thus

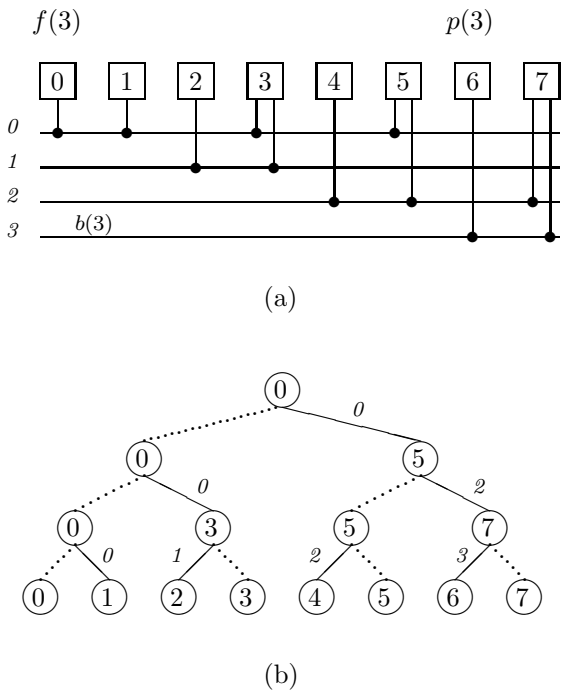

Figure 1. Running $Bin(3)$ on an 8×4 MBN

$Bin(n)$ requires at least n steps as the height of $\mathcal{T}(n)$ is n; on the other hand, particular reduction operations (such as finding the OR of 2^n bits) can be performed in constant time on certain models. We will loosely use the term "binary-tree MBN" to refer to an MBN suitable for running binary-tree algorithms.

To run $Bin(n)$ on a network with 2^n processors, each of the $2^{n+1} - 1$ nodes of $\mathcal{T}(n)$ is mapped to one of the 2^n processors of the network. In Figure 1(b), the numbers within circles denote processors. Edges represent communications and are labeled with the buses used for them. Edges whose end-points are mapped to the same processor do not require a communication and are shown dotted.

In running a binary-tree algorithm on an MBN, we assume that in one step a processor can read from or write on each bus it is connected to, and perform an internal operation using operands from its local memory or input ports. This assumption is not unreasonable as each processor in the MBNs considered in this paper has at most two ports.

Throughout this paper, we assume that $Bin(n)$ is run on a $2^n \times 2^{n-1}$ MBN; there is no loss of generality in this assumption.

3. Lower Bound for Optimal MBNs

In this section we show that any degree-2 MBN that runs $Bin(n)$ optimally has a loading of $\Omega(n^{\frac{2}{3}})$. This result is a non-trivial tightening of the $\Omega(n^{\frac{1}{2}})$ lower bound established previously [3].

Consider a $2^n \times 2^{n-1}$ MBN, $X(n)$, that runs $Bin(n)$ optimally in n steps and whose degree and loading are 2 and $L \leq n$, respectively. For any $1 \leq s \leq n$, let $X_s(n)$ denote a $2^n \times 2^{n-1}$ MBN, that includes only those connections of $X(n)$ that are used in at least one of the steps $1, 2, \cdots, s$. Let $X_0(n)$ be a $2^n \times 2^{n-1}$ MBN with no connections. Clearly $X_n(n) = X(n)$.

For any $0 \leq s \leq n$, a processor of $X_s(n)$ that has a degree of 2 is said to be a *full processor of step s*. A processor holding a partial result or an input at the end of step s (or at the beginning of step $s+1$) is called a *result processor of step s*; otherwise it is a *non-result processor of step s*. The following fact is used throughout this section without explicit mention.

Fact 1 [3] *For any $L \leq s \leq n$, each result processor of step s is a full processor of step s.* ■

A bus is said to be *active in step s* (where $s \geq L$) iff the bus is connected to at least one result processor of step s. The following result shows that the pool of active buses that the MBN uses to combine intermediate results cannot increase; as a result, connections due to steps toward the end of the algorithm are confined to a small number of buses.

Fact 2 [3] *For any $s \geq L$, if a bus is active in step $s+1$, then it is active in step s.* ■

To keep track of connections (that are counted towards the lower bound), each connection of $X_s(n)$ is assigned to a processor of the MBN (the processor is said to *own* the connection [3]). This assignment is such that (1) each connection is owned by at most one processor (therefore, no connection is counted more than once), (2) all connections owned by a processor are to one of the two buses to which the processor is connected; this ensures that all connections owned by a processor are distributed among two buses.

Consider a processor p connected to buses b and b'. Besides p, let buses b and b' be connected to processors $p_1, p_2, \cdots, p_\ell$ and $q_1, q_2, \cdots, q_{\ell'}$, respectively, where $\ell, \ell' < L$. Processors $p_1, p_2, \cdots, p_\ell, q_1, q_2, \cdots, q_{\ell'}$ are collectively called *neighbors* of p; these are processors reachable from p by a single bus hop. Also, for integers i, j with $i \leq j$, let $[i, j]$ denote the set (interval) $\{i, i+1, \cdots, j-1, j\}$.

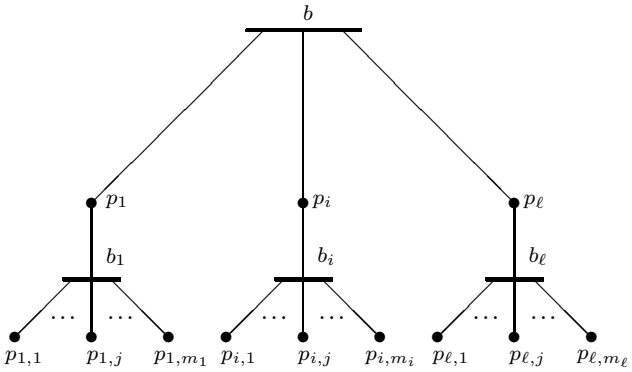

Figure 2. Processors and buses in the neighborhood of bus b

Lemma 1 *For $s > L$, if p is not a result processor of step $s-1$, and is a result processor of steps $s, s+1, \cdots, s+x-1$ (for some $x > 0$), then the following assertions hold:*
(i) For each step s' in the interval $[L, s-1]$, at least $x+1$ neighbors of p are result processors of step s'.
(ii) At the end of step $s+x-1$, the neighbors of p collectively own at least $\left((x+1)(s-L) + \frac{x(x-1)}{2}\right)$ connections.

Proof: Omitted for brevity. ■

Now we are in a position to prove the main result of this paper.

Theorem 2 *For any degree-2, $2^n \times 2^{n-1}$ MBN that runs $Bin(n)$ optimally in n steps, its loading is $\Omega(n^{\frac{2}{3}})$.*

Proof: Let the loading of the MBN be $L \leq n$. By Fact 2, there is a bus that is active during each of steps in the interval $[L, n]$; let this bus be b. Let full processors $p_1, p_2, \cdots, p_\ell$ (for some $\ell \leq L$) be connected to bus b. For each i (where $1 \leq i \leq \ell$), let processor p_i be connected to bus b_i (in addition to bus b). Besides processor p_i, let bus b_i be connected to $m_i < L$ processors (see Figure 2).

For any given step $s \in [L, n]$, at least one of $p_1, p_2, \cdots, p_\ell$ is a result processor (as b is active during each step of $[L, n]$). Therefore, the interval $[L, n]$ can be partitioned into k subintervals, $I_1, I_2, \cdots, I_k$, as follows:

- In each step of subinterval I_j, processor $\pi_j \in \{p_i : 1 \leq i \leq \ell\}$ is a result processor.
- For $j > 1$, $\pi_j \neq \pi_{j-1}$.

Let interval I_j be of length x_j (where $x_j \in [1, n-L+1]$). Clearly, $\sum_{j=1}^{k} x_j = n - L + 1$. Also $I_1 = [L, L + x_1 - 1]$ and for $j > 1$, $I_j = \left[L + \sum_{r=1}^{j-1} x_r \ , \ L - 1 + \sum_{r=1}^{j} x_r \right] = [s_j, s_j + x_j - 1]$ (say).

Applying Lemma 1 to I_j, the number of connections owned by neighbors of π_j (at the end of step $s_j + x_j - 1$) is

$$(x_j + 1)(s_j - L) + \frac{x_j(x_j - 1)}{2}$$

Summing this for all k intervals we can assert that the number of connections collectively owned by processors $p_1, p_2, \cdots, p_\ell$ and their neighbors is at least

$$\sum_{j=1}^{k} \left((x_j + 1)(s_j - L) + \frac{x_j(x_j - 1)}{2} \right)$$

which can be shown to be $\Theta((n - L)^2) = \Theta(n^2)$. These connections are distributed among $1 + \ell + \sum_{i=1}^{\ell} m_i \leq 1 + L + L(L-1)$ buses connected to the above processors (see Figure 2). Since each bus can have at most L connections we have

$$L \left(1 + L + L(L-1) \right) = \Theta(L^3) = \Omega(n^2)$$

which implies that $L = \Omega(n^{\frac{2}{3}})$. ∎

Remark: The results of this section also hold for MBNs with more than 2^{n-1} buses.

4. Loading-Speed Tradeoff

Let $2^{h(L)}$ be the size of the largest instance of a binary-tree algorithm that can be run optimally in $h(L)$ steps on a degree-2, loading-L MBN. From Theorem 2 we know that $h(L) = O(L^{\frac{3}{2}})$. The degree-2, $\Theta(n)$-loading optimal MBN for $Bin(n)$ presented in [15] gives the bound $h(L) = \Omega(L)$.

Theorem 3 *Any degree-2, loading-L $2^n \times 2^{n-1}$ MBN requires at least $n + \left\lfloor \frac{n}{h(L)+1} \right\rfloor$ steps to run $Bin(n)$.*

Proof: Since the MBN runs $Bin(n)$ suboptimally, there are some nodes of $\mathcal{T}(n)$ with "delays" in them. A node with delay δ, passes its value (input/partial result) to its parent, δ steps after this value is available to it. This delay may be use to transfer the value to a different processor with less demands in its buses.

The definition of $h(L)$ ensures that for $x > h(L)$, tree $\mathcal{T}(x)$ has at least one delayed node in it; therefore $Bin(x)$ requires at least $x + 1$ steps on the given MBN.

Now divide $\mathcal{T}(n)$ into $\left\lceil \frac{n}{h(L)+1} \right\rceil$ parts, P_i, where $1 \leq i \leq \left\lceil \frac{n}{h(L)+1} \right\rceil$. Each part, P_i, (except perhaps the last one) consists of $h(L) + 1$ contiguous levels of $\mathcal{T}(n)$. That is, these parts contains trees isomorphic to $\mathcal{T}(h(L) + 1)$ that must each contain at least one delayed node (with delay $\delta \geq 1$). For part P_1 that starts from the leaves, the roots of the $\mathcal{T}(h(L) + 1)$'s in this part obtain the values no earlier than at step $h(L) + 2$. As a result, the roots of the $\mathcal{T}(h(L) + 1)$'s in the next part P_2 obtain their values no earlier than at step $2(h(L) + 2)$. In general for $1 \leq i \leq \left\lfloor \frac{n}{h(L)+1} \right\rfloor$, the roots of the $\mathcal{T}(h(L) + 1)$'s of part P_i obtain their values no earlier than at step $i(h(L) + 2)$. Without the delayed nodes, these roots would have obtained their values at step $i(h(L) + 1)$, so the additional time taken is at least i. Therefore, the additional time (due to delayed nodes) taken before the the root of $\mathcal{T}(n)$ obtains its value is at least $\left\lfloor \frac{n}{h(L)+1} \right\rfloor$. ∎

When the loading L is a constant, Theorem 3 requires the additional time (beyond the minimum required n steps) to run $Bin(n)$ to be $\Omega(n)$. This is because $h(L) = O(L^{\frac{3}{2}}) = O(1)$, for constant L. We now show that this lower bound on the additional time is tight by presenting a degree 2, loading-4 MBN that runs $Bin(n)$ in $2n-3$ steps (that is with $n-3$ additional steps).

Consider a degree-2, constant loading-ℓ, $2^n \times 2^{n-1}$ MBN, $M(n)$ that has the following properties.

- The processor $f(n)$ that holds the final result of $Bin(n)$ has a degree of 1; that is, $f(n)$ is connected to only one bus.
- There is a bus $b(n)$ with loading $\ell - 2$; that is, two more processors could be connected to bus $b(n)$, without increasing the loading of $M(n)$.
- One of the processors, $p(n)$, connected to bus $b(n)$ has degree 1; that is, processor $p(n)$ is not connected to any bus other than $b(n)$.

Processors $f(n)$, $p(n)$ and bus $b(n)$ will be called the special elements of $M(n)$. An example of such an MBN is the 8×4 MBN shown in Figure 1(a). For this MBN the loading ℓ is 4, and $f(3)$, $p(3)$ are processors 0 and 6, respectively, while bus *3* is $b(3)$. Figure 1(b) shows how $Bin(3)$ is run on $M(3)$ in 3 steps It is easy to verify, that $M(3)$ possesses the above properties.

We now show how two copies of $M(n)$ can be used to construct the $2^{n+1} \times 2^n$ MBN, $M(n+1)$. To distinguish these copies, we name them $M_1(n)$ and $M_2(n)$ and refer to their special elements as $f_i(n)$, $p_i(n)$ and $b_i(n)$ for $i = 1, 2$. To construct $M(n+1)$ from $M_1(n)$ and $M_2(n)$,

all that needs to be done is to connect processors $f_1(n)$ and $f_2(n)$ to bus $b_1(n)$. Designate $p_1(n)$ to be $f(n+1)$, $p_2(n)$ to be $p(n+1)$ and $b_2(n)$ to be $b(n+1)$.

MBN $M(n+1)$ has a loading of ℓ as the only two added connections are to bus $b_1(n)$ that has only $\ell-2$ connections to start with. Its degree is 2 as the added connections are one each from processors $f_1(n)$ and $f_2(n)$, that had only one connection each to begin with. It is easy to verify that each of the processors $f(n+1) = p_1(n)$ and $p(n+1) = p_2(n)$ has degree 1 and that bus $b(n+1) = b_2(n)$ has loading $\ell-2$. If we can now establish that $f(n+1)$ holds the final result, then $M(n+1)$ will satisfy the three properties stated above for $M(n)$.

When $Bin(n+1)$ is run on $M(n+1)$, the first n levels (starting from the leaves) are run simultaneously on $M_1(n)$ and $M_2(n)$; the results of these steps are in processors $f_1(n)$ and $f_2(n)$. These processors use bus $b_1(n)$ to send the (partial) results to $f(n+1) = p_1(n)$. (Recall that processors $f_1(n)$ and $f_2(n)$ have been connected to bus $b_1(n)$, while $f(n+1) = p_1(n)$ is already connected to this bus.) Processor $f(n+1)$ now computes the final value of $Bin(n+1)$. From this discussion it should also be evident that if $M(n)$ runs $Bin(n)$ in $T(n)$ steps, then $M(n+1)$ runs $Bin(n+1)$ in $T(n+1) = T(n)+2$ steps. This is because both processors $f_1(n)$ and $f_2(n)$ use the same bus, $b_1(n)$, to send their partial results to processor $f(n+1)$; this introduces a delay. Coupled with the fact that $T(3) = 3$ (see Figure 1), this gives $T(n) = 2n-3$, for $n \geq 3$.

Thus we have the following result.

Theorem 4 *For any $n \geq 3$, the degree-2, loading-4, $2^n \times 2^{n-1}$ MBN, $M(n)$, runs $Bin(n)$ in $2n-3$ steps.* ■

The results of this section show that a degree-2, constant loading MBN can run $Bin(n)$ if and only if it is permitted to take $\Theta(n)$ steps more than the optimal.

5. Concluding Remarks

We have proved that for any degree-2 MBN that runs $Bin(n)$ optimally, its loading is $\Omega(n^{\frac{2}{3}})$. We have also shown a trade-off between the speed and loading of degree-2 MBNs. In particular we have proved that a degree-2, constant loading MBN can run $Bin(n)$ if and only if it is permitted to take $\Theta(n)$ steps more than the optimal.

Open problems include narrowing the gap between the lower bound ($\Omega(n^{\frac{2}{3}})$) and upper bound ($O(n)$) on the loading of optimal, degree-2 MBNs. A similar gap also exists for MBN permitted to run $Bin(n)$ with delays.

References

[1] V. Bokka, H. Gurla, S. Olariu, L. Swing, and L. Wilson, "Time-Optimal Domain Specific Querying on Enhanced Meshes," *IEEE Trans. Parallel & Distributed Systems*, **8**, 1997, pp. 13–24.

[2] S. Cheung and F. C. M. Lau, "Routing with Locality on Meshes with Buses," *J. Parallel & Distributed Computing*, **33**, 1996, pp. 84–90.

[3] H. P. Dharmasena and R. Vaidyanathan, "An Optimal Multiple Bus Network for Fan-in Algorithms," *Proc. International Conference on Parallel Processing*, 1997, pp. 100–103.

[4] O. M. Dighe, R. Vaidyanathan and S. Q. Zheng, "The Bus-Connected Ringed Tree: A Versatile Interconnection Network," *J. Parallel & Distributed Computing*, **33**, 1996, pp. 189–196.

[5] S. Fujita and M. Yamashitar, "Fast Gossiping on Mesh-Bus Computers," *IEEE Trans. Computers*, **45**, 1996, pp. 1326–1330.

[6] Z. Guo and R. G. Melhem, "Embedding Binary X-Trees and Pyramids in Processor Arrays with Spanning Buses," *IEEE Trans. Parallel & Distributed Systems*, **5**, 1994, pp. 664–672.

[7] J.-w. Jang, M. Nigam, V. K. Prasanna, and S. Sahni, "Constant Time Algorithms for Computational Geometry on the Reconfigurable Mesh," *IEEE Trans. Parallel & Distributed Systems*, **8**, 1997, pp. 1–12.

[8] S. T. Kamath and R. Vaidyanathan, "Running Weak Hypercube Algorithms on Multiple Bus Networks," *Proc. ISCA International Conference on Parallel & Distributed Computing Systems*, 1997, pp. 217–222.

[9] P. Kulasinghe and A. El-Amawy, "On the Complexity of Bussed Interconnections," *IEEE Trans. Computers*, **44**, 1995, pp. 1248–1251.

[10] P. Kulasinghe and A. El-Amawy, "Optimal Realization of Sets of Interconnection Functions on Multiple Bus Systems," *IEEE Trans. Computers*, **45**, 1996, pp. 964–969.

[11] F. T. Leighton, *Introduction to Parallel Algorithms and Architectures: Arrays · Trees · Hypercubes*, Morgan Kaufmann Publishers, San Mateo, CA, 1992.

[12] K. Nakano, "A Bibliography of Published Papers on Dynamically Reconfigurable Architectures," *Parallel Processing Letters*, **5**, 1995, pp. 111–124.

[13] K. Nakano, S. Olariu and J. L. Schwing, "Broadcast Efficient Algorithms on the Coarse-Grain Broadcast Communication Model with Few Channels," *Proc. International Parallel Processing Symposium*, 1998, pp. 31–35.

[14] M. J. Serrano and B. Parhami, "Optimal Architectures and Algorithms for Mesh-Connected Parallel Computers with Separable Row/Column Buses," *IEEE Trans. Parallel & Distributed Systems*, **4**, 1993, pp. 1073–1080.

[15] R. Vaidyanathan and A. Padmanabhan, "Bus-Based Networks for Fan-in and Uniform Hypercube Algorithms," *Parallel Computing*, **21**, 1995, pp. 1807–1821.