To appear in *Mathematical and Computational Modelling*

Shift-Product Networks

Marilynn Livingston
Computer & Information Science
University of Oregon
Eugene, OR 97403-1202

Quentin F. Stout*
Elec. Eng. and Comp. Sci.
University of Michigan
Ann Arbor, MI 48109-2122

Abstract

Economics is the principle driver of current trends to use commodity components in the construction of parallel systems for a range of sizes. One objective is to enable the user to purchase a small system initially, and then extend it through a range of sizes as needs dictate. A desirable feature is to have good system performance through the range of sizes; an undesirable feature is to require the user to purchase excess hardware that will not be used until the system is grown to its maximum allowable size. In this paper, we give a new construction which reconciles these two conflicting factors by introducing a way to interconnect several components of a given small network using only the routers for the small network, with one additional port per router. This construction, which we call *shift-product*, does not unduly raise the communication diameter of the resulting large network.

Key Words: interconnection networks, scalable networks, shuffle-exchange networks, Cayley graphs, product graphs, de Bruijn network.

1 Introduction

Designers of interconnection networks for parallel computers must face the economic fact that the vast majority of systems sold will be quite small, and that the users of such systems do not want to pay for over-designed systems. However, some users are interested in having systems that can scale to larger sizes, with reasonably good performance throughout the range of sizes. Many different approaches to this have been investigated, and several have been used to construct real machines. For example, the SGI Origin components are used to construct the Cray Origin, Hewlett Packard SMPs are used to make Convex machines, and IBM SMPs will soon be available as nodes in the IBM SP series. In these examples, a component involves approximately 4–8 processors along with memory and I/O devices.

The problem that we consider here is:

> How can one use a commodity small system as a building block, plus small additional cost, to produce a parallel computer with good performance for a range of system sizes?

"Good performance" is a complex criterion, incorporating characteristics of the interconnection network such as diameter, bisection bandwidth, fault tolerance, ease of routing, and so forth. "Small additional cost" is often measured by factors such as the extra hardware to enable system growth, number of extra communication links per node, design cost, etc. In theory, the "range of system sizes" should be infinite, but in practice, it may be most important to focus on systems which can be grown to at most an order or two of magnitude bigger than the component system.

In this paper, we describe a method of construction which uses as building block a fixed, but arbitrary, component system. With one unused input and output port per processor, it interconnects the component systems to yield a large system with relatively small diameter, and maintains the attractive properties of the original network. Our construction is called a shift-product.

*Partially supported by NSF grant DMS-9504980.

In Section 2 we survey some previously examined methods for constructing large systems from small ones. In Section 3 we define the shift-product construction, and examine some of its properties. In Section 4 we compare the shift-product to some related constructions, and show how to modify the shift-product to produce some intermediate-sized networks. In Section 5 we provide some conclusions.

2 Some Methods of Construction

There are many different approaches to network design, the common goal being to produce a network with provably good performance. Because of the many conflicting factors which affect performance, such as small diameter and low node degree, simplicity of routing protocols and fast routing, high bandwidth and low cost, various tradeoffs must be studied and evaluated. When the network design has a formal description, its properties can be more readily established. Moreover, formal methods of construction can often be guided by a specification of the network properties desired. We will note several examples of this in this and the next section.

A most basic composition of networks is that of the direct product. Specifically, in composing networks, one would like to maintain desirable network properties such as symmetry, low degree, diameter, scalability, and efficient mappings of algorithms and embeddings of certain classes of networks. The behavior of several of these properties under the direct product composition was studied in [2].

2.1 Direct Product

Let G and H denote two graphs. The *direct product* of G by H, denoted by $G \times H$, is the graph whose set of nodes is the Cartesian product of the set of nodes of G by the set of nodes of H, and whose edges are determined as follows: if $n_1 = (g_1, h_1)$ and $n_2 = (g_2, h_2)$ are two nodes in $G \times H$, then (n_1, n_2) is an edge provided that either

(i) $h_1 = h_2$ and (g_1, g_2) is an edge in G, or

(ii) $g_1 = g_2$ and (h_1, h_2) is an edge in H.

It is easy to check that the degree of a node (n_1, n_2) is the sum of the degrees of n_1 and n_2 and the diameter of $G \times H$ is the sum of the diameters of G and H. The product construction extends naturally to more than two factors.

Many of the networks actually used in parallel computers can be expressed as the direct product of simpler networks. For example, the k-dimensional mesh is a direct product of k paths, the k-dimensional hypercube is the direct product of k paths of length two, and the k-dimensional torus is a direct product of k cycles.

Routing in a direct product of networks often makes use of the existing routing procedures in each of the factor networks, in some hierarchical order. For example, routing from (x_1, x_2) to (y_1, y_2) may use the routing of the first factor network to move to (y_1, x_2) and then the routing of the second factor to move to (y_1, y_2).

To use the direct product construction as a means of upgrading a commodity system S through systems S, $S \times S$, $S \times S \times S, \ldots$ would result in increasing the system by a factor of $|S|$ at each upgrade while increasing the node degree by one at each stage. The growth of node degree is a major disadvantage of this particular approach.

Choosing a cycle (ring) topology for one of the factors helps to maintain a low degree as upgrades are constructed, independent of the size of the system. For example, upgrades would be of the form $S \times C_{n_1}$, $S \times C_{n_2}, \ldots, S \times C_{n_t}$, where C_{n_i} is a cycle with n_i nodes, and $n_1 < n_2 < \ldots < n_t$. This type of interconnection network is used in the Convex Exemplar and in the Cray Origin. The construction used in the Convex 2000 series can be characterized as the direct product of S and T, where S is a complete graph and T is a two-dimensional torus of degree 4.

The hypercube topology, which is a direct product construction, has many attractive features, but suffers from the drawback of high node degree since the degree grows with each new factor.

In several proposed networks, the hypercube topology is introduced with some variation to control the node degree. An example of this approach is the cube-connected cycles (CCC) [7]. We will discuss several of these networks in Section 2.2.

2.2 A Hierarchical Product

Many recent network constructions can be viewed as a hierarchical structure in which a copy of a fixed graph is embedded at each node of a host network [1, 4, 5, 7]. We offer a formal description of this method of composition and call it a *hierarchical product*. Start with a graph G of g vertices, and a graph H where each of its nodes has indegree and outdegree no more than g. Assume that the edges of H are labeled with a vertex in G, so that at each vertex of H, no two incoming edges have the same label, and no two outgoing edges have the same label. The hierarchical product of G and H, denoted $G \triangle H$, has vertices (u, v) for all $u \in G$ and $v \in H$. In $G \triangle H$, there is an edge from (u_1, v_1) to (u_2, v_2) if and only if

(i) $v_1 = v_2$ and (u_1, u_2) is an edge in G, or

(ii) $u_1 = u_2$ and (v_1, v_2) is an edge in H which is labeled u_1.

Strictly speaking, we should have the labeling, λ, of the edges of H appear in the notation for hierarchical product. In cases where the labeling needs to be explicit we will use the more precise notation $G \triangle_\lambda H$. As in the case of direct product, there is a natural extension of the hierarchical product to more than two factors.

Note that in $G \triangle H$, nodes of the form $(*, v)$, for $v \in H$, form a subgraph which is a copy of G. These subgraphs are called *supernodes*. Furthermore, in the case of G a single node, (or, if we "shrink" G down to a single node) we recognize $G \triangle H$ as a copy of H.

We can view the CCC network as a hierarchical product. In the CCC of dimension n, G is a cycle of n processors and H is an n-dimensional hypercube. The vertices of G are labeled $0 \ldots g-1$ in circular fashion, and the label on an edge in H is the dimension that it traverses. Another example based on hypercubes is the *cube-connected cubes* network, which is the hierarchical product of two hypercubes $G = Q_d$ and $H = Q_n$, where $2^d = n$. Again, label the edges of H by the dimension they traverse, and the vertices of G are labelled in the standard hypercube fashion.

Historically, CCCs were promoted as a solution to the "commodity" problem because the nodes have fixed degree 3, i.e., the same node can be used in all CCCs, no matter how large, and yet the CCC is nearly as useful as the hypercube which has a degree which grows logarithmically with the system size. However, the commodity here was too small, being merely a single node (processor). In cube-connected cubes, the commodity is the supernode G, rather than a commodity node. Moving up to larger commodities, such as boards or cabinets, provides significantly greater economic benefits over commodity processors.

In both the CCC and the cube-connected cube examples, for each vertex of H, every vertex of G appears as an edge label. In Section 4, we will give examples where not all vertices of G appear as edge labels of H.

In the next section we will introduce the shift-product, which is a particular hierarchical product in which the de Bruijn network [3, 6, 8] is a factor.

3 Shift-Products

Let S be an arbitrary connected graph and let k be an integer > 1. The k-th *shift product* of S, denoted by $\bowtie S^k$, is a graph whose nodes consist of k-tuples of nodes of S, and there is an edge from node $a = a_1 a_2 \ldots a_k$ to node $b = b_1 b_2 \ldots b_k$ provided:

(i) $a_i = b_i$ for $2 \le i \le k$ and there is an edge in S from a_1 to b_1 (edge of type S), or

(ii) $b_i = a_{i+1}$ for $1 \le i \le k-1$ and $b_k = a_1$ (a shift arc)

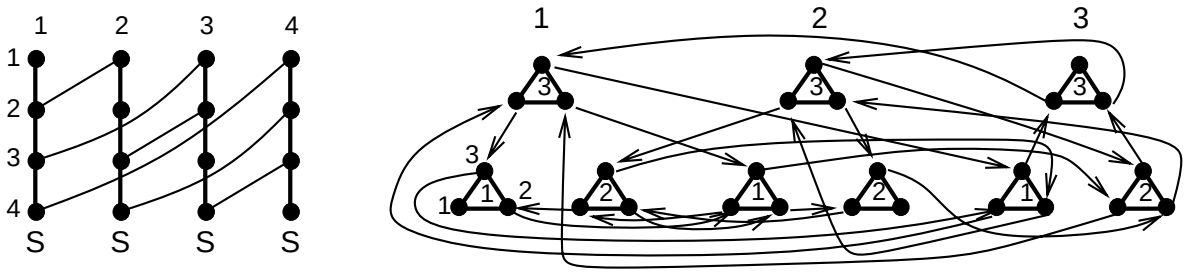

Figure 1: Shift-products: $\bowtie (P_4)^2$ and $\bowtie (C_3)^3$

The type S edges gives us $|S|^{k-1}$ copies of the graph S, namely all subgraphs of the form $*a_2 \ldots a_k$. The edges between these supernodes are provided by the "shift" arcs. Figure 3 shows the construction of $\bowtie S^2$ when S is a path of length 4, and the construction of $\bowtie S^3$ when S is a cycle of length 3. The 2-dimensional shift-product might well be called a transpose network, since nodes ab and ba are connected. The shift arcs in a 2-dimensional shift-product are bidirectional, but in general are only unidirectional for all higher dimensions.

From the definition, one can immediately conclude that $\bowtie S^k$ has $|S|^k$ nodes, and that a node $a_1 a_2 \ldots a_k$ in $\bowtie S^k$ has incoming degree and outgoing degree one more than the corresponding degree of node a_1 in S. By comparison, the k-fold direct product of S also has $|S|^k$ nodes, but the maximum incoming or outgoing degree of a node is k times the maximum degree of a node in S.

To route in $\bowtie S^k$ from node $a_1 a_2 \ldots a_k$ to node $b_1 b_2 \ldots b_k$:

(i) use the S routing to go from $a_1 a_2 \ldots a_k$ to $a^{[2]} = b_2 a_2 \ldots a_k$

(ii) use a shift arc to move to $a^{[3]} = a_2 \ldots a_k b_2$

(iii) use the S routing to move to $a^{[4]} = b_3 a_3 \ldots a_k b_2$

(iv) use a shift arc to move to $a^{[5]} = a_3 \ldots a_k b_2 b_3$

(v) continue, alternating a move in S followed by a shift arc to reach $a^{[2k-1]} = a_k b_2 b_3 \ldots b_k$

(vi) use the S routing to move to $b_1 b_2 b_3 \ldots b_k$.

We see that at most k moves along S-edges and at most $k-1$ moves along shift arcs will be needed, yielding a diameter of no more than $dk + k - 1$, where d is the diameter of S. To see that the diameter is no smaller than this, let x and y be two nodes in S at distance d. Straightforward analysis shows that the node $x_1 x_2 \ldots x_k$, where $x_i = x$, is at distance $dk+k-1$ from node $y_1 y_2 \ldots y_k$, where $y_i = y$. In contrast, the diameter of the k-fold direct product is dk. Thus the shift-product has similar diameter to the k-fold direct product, but has a significantly smaller degree which does not grow with k.

In addition to good performance through a wide range of system sizes, it is (commercially) desirable to have several intermediate sizes available in the upgrade process. Composition of networks using factor networks limits the intermediate sizes available for most choices of the factor networks. In a direct product where all factors are a commodity system S, the system sizes must grow by a factor of $|S|$. With the hybrid ring product $S \times C_n$, things look more attractive because the system can increase by the additive amount $|S|$. Unfortunately the diameter of such a system grows linearly with the number of supernodes. Product networks that involve a mesh or a hypercube can only increase by very specific quantities: some multiple of a linear dimension of the mesh, doubling the size of the hypercubes in the cube-connected cubes, for example, are our only alternatives unless we want to lose the symmetry and change the routing. This would involve the use of "faulty" or "incomplete" systems. There have many numerous studies for both the mesh and the hypercube networks from this point of view. A more recent study involved using small meshes as basic building blocks to form larger incomplete meshes [10]. Such systems require significant changes to the basic building block and routing networks, in general.

The nature of the shift-product will allow us to realize many more intermediate system sizes, as we show in the next section.

4 Modified Shift-Products

The shift-product $\bowtie S^k$ can be viewed as a hierarchical product of S and a $(k-1)$-dimensional de Bruijn graph over the alphabet of the nodes of S. A j-dimensional de Bruijn graph over the alphabet A has $|A|^j$ nodes, where the index of each node is of the form $a_1a_2\ldots a_j$, where each a_i is in A. Node $a_1a_2\ldots a_j$ has an edge to all nodes of the form $a_2a_3\ldots a_jb$, for all $b \in A$.

Viewed in this manner, there is a natural way to create smaller k-dimensional shift-products. Namely, let A be any subset of nodes of S, and utilize only supernodes which form a $(k-1)$-dimensional de Bruijn network over A. That is, supernodes of the form $*a_2a_3\ldots a_k$, for $a_i \in A$. The number of nodes in this modified shift-product is $|S| \cdot |A|^{k-1}$. It is easy to verify that the normal routing algorithm performs perfectly in this subgraph, in that no attempt is ever made to route to a supernode not in the subgraph. In each supernode, there will be some nodes which have no connections outside of the supernode.

Yet a further modification can be had by noticing that in the j-dimensional de Bruijn network over A, nodes of the form $aa\ldots a$ need never arise as intermediate nodes, as any subset of such nodes can be removed without increasing the diameter. However, to exploit this, a slight change needs to be made to the routing previously described. For example, in $\bowtie \{0,1\}^3$, to route from 010 to 001 would follow the steps

$$010 \xrightarrow{S} 010 \xrightarrow{\text{shift}} 100 \xrightarrow{S} 100 \xrightarrow{\text{shift}} 001,$$

where the label above the arrow indicates the type of edge used. Note that for completeness we have included self-loops. In this path, an intermediate node of the form *00 is used, so had this supernode been removed, the routing would have failed. However, it was not necessary to utilize this as an intermediate node, since the path

$$010 \xrightarrow{S} 110 \xrightarrow{\text{shift}} 101 \xrightarrow{S} 001,$$

avoids any intermediate node of the form $*aa$.

One way to remove all such nodes as intermediate stages in the routing is to notice that whenever the standard routing would create

$$a*a\ldots a \xrightarrow{\text{shift}} *a\ldots a \xrightarrow{S} ba\ldots a \xrightarrow{\text{shift}} a\ldots ab,$$

it can replaced by

$$a*a\ldots a \xrightarrow{S} b*a\ldots a \xrightarrow{\text{shift}} *a\ldots ab.$$

This maintains proper routing, and does not increase the diameter.

To illustrate how much these constructions expand the number of systems available, suppose that the component system S has 8 nodes. Then $\bowtie S^2$ has 8 supernodes (64 nodes), and $\bowtie S^3$ has 64 supernodes (256 nodes). These are the only systems available with 64 or fewer supernodes, if the standard shift-product (or the standard direct product) construction is used. However, using the above subgraph constructions, one can create subgraphs of $\bowtie S^2$ which have any number of supernodes from 1 to 8, and subgraphs of $\bowtie S^3$ which have 1, 2 $(= 2^2-2)$, 3 $(= 2^2-1)$, 4 $(= 2^2)$, 6 $(= 3^2-3)$, 7 $(= 3^2-2)$, 8 $(= 3^2-1)$, 9 $(= 3^2)$, 12 $(= 4^2-4)$, ...16, 20, ..., 25, 30, ..., 36, 42, ...49, 56, ...64 supernodes. This gives one significant flexibility in building a system of the correct size.

5 Conclusion

We have introduced the "hierarchical product" of networks, and highlighted a special case of it, namely the shift-product. The shift-product construction, together with its modifications, illustrates a practical method of constructing parallel systems from commodity small components. Earlier efforts along this line tended to emphasize commodity nodes (processors), as opposed to our use of commodity supernodes (systems). Our construction requires only one extra input and

output port per node. Upgrading is easy, requiring only small changes in the operating system to update the hardware additions.

The systems so constructed are scalable, of fixed degree and low diameter, through a wide range of sizes. Furthermore, as long as the commodity components are balanced with respect to compute power, memory, and I/O, the upgraded system will be balanced, too.

References

[1] G. E. Carlsson, J. E. Cruthirds, H. B. Sexton, and C. G. Wright, "Interconnection networks based on a generalization of cube-connected cycles", *IEEE Trans. Computers*, Vol. C-34, No. 8, (1985) 769–772.

[2] K. Day and A.-E. Al-Ayyoub, "The cross product of interconnection networks", *IEEE Trans. Parallel and Distrib. Systems*, Vol. 8, No. 2, (1997) 109–118.

[3] N. G. de Bruijn, "A combinatorial problem", *Nederl. Akad. Wetensch. Proc.* Ser. A 49 (1946) 758–764.

[4] G.-H. Chen and H.-L. Huang, "Cube-connected modules: a family of cubic networks", *Int. Symp. Par. Architectures, Algorithms and Networks* (1994) 57–64.

[5] K. Ghose and K. R. Desai, "Hierarchical cubic networks", *IEEE Trans. Parallel and Distrib. Systems*, Vol. 6, No. 4 (1995) 427–435.

[6] D. F. Hsu and D. S. L. Wei, "Permutation routing and sorting on directed de Bruijn networks", *Int. Conf. on Parallel Processing*, I, (1995) 96–100.

[7] F. P. Preparata and J. E. Vuillemin, "The cube-connected cycles: a versatile network for parallel computation", *Comm. ACM*, Vol. 24 (1981) 300–309.

[8] M. R. Samatham and D. K. Pradhan, "The deBruijn multiprocessor network: a versatile parallel processing and sorting network for VLSI", *Interconnection Networks for High-Performance Parallel Computers*, IEEE Computer Society Press (1994) 153-167.

[9] S. T. Schibell and R. M. Stafford, "Processor interconnection networks from Cayley graphs", *Discrete Applied Math.*, Vol. 40 (1992) 333–357.

[10] M. Yang and L. M. Ni, "Incremental design of scalable interconnection networks using basic building blocks", *Proc. 7th IEEE Symp. Parallel and Distrib. Processing* (1995) 252–259.