Products of Networks with Logarithmic Diameter and Fixed Degree

Kemal Efe, *Member, IEEE*, and Antonio Fernández

Abstract—**This paper first analyzes some general properties of product networks pertinent to parallel architectures and then focuses on three case studies. These are products of complete binary trees, shuffle-exchange, and de Bruijn networks. It is shown that all of these are powerful architectures for parallel computation, as evidenced by their ability to efficiently emulate numerous other architectures. In particular, r-dimensional grids, and r-dimensional meshes of trees can be embedded efficiently in products of these graphs, i.e. either as a subgraph or with small constant dilation and congestion. In addition, the shuffle-exchange network can be embedded in r-dimensional product of shuffle exchange networks with dilation cost $2r$ and congestion cost 2. Similarly, the de Bruijn network can be embedded in r-dimensional product of de Bruijn networks with dilation cost r and congestion cost 4. Moreover, it is well known that shuffle-exchange and de Bruijn graphs can emulate the hypercube with a small constant slowdown for "normal" algorithms. This means that their product versions can also emulate these hypercube algorithms with constant slowdown. Conclusions include a discussion of many open research areas.**

Index Terms—**Product networks, interconnection networks, parallel architectures, multiprocessors, graph embedding, application specific array processors, emulation, embedded architectures.**

I. Introduction

NETWORKS[1] with small diameter, small vertex degrees, and large bandwidth are well suited for massively parallel computation. The hypercube is a well known example of a network with small diameter and large bandwidth, but the vertex degree of hypercube grows logarithmically with the number of vertices, making it hard to build scalable architectures. Grids have larger diameter than hypercubes, but due to their fixed and small vertex degrees their popularity has been increasing in recent years. A small vertex degree implies that the system can be implemented with a small hardware cost spent for the communication channels. A fixed vertex degree implies that the system can be expanded without having to modify the individual nodes. Although these are important advantages offered by the grid network, besides its ability to efficiently compute certain classes of algorithms, it is not well suited for other classes of computations, including divide-and-conquer, ascend-descend, parallel merge, etc.

1. We use the terms "graph" and "network" interchangeably.

Manuscript received Mar. 8, 1993; revised Dec. 10, 1993.

K. Efe is with the Center for Advanced Computer Studies, University of Southwestern Louisiana, Lafayette, LA 70504; e-mail: efe@bayour21.cacs.usl.edu.

A. Fernández is with the Departamento de Arquitectura y Tecnología de Computadores at the Universidad Politécnica de Madrid, Spain.

IEEECS Log Number D95036.

We present product networks as generalizations from grids and show that the presented networks preserve the fixed degree property of grids while improving several of its properties, such as reducing the diameter to logarithmic values as well as increasing the bandwidth. Moreover, we show that every product network can emulate grids with small overhead and also offer several other advantages depending on the factor graph used.

In simple terms, the r-dimensional product of N-node graph G is obtained from the r-dimensional N^r-node grid by replacing the linear connections of the grid for the interconnection pattern of G. As an example, Fig. 1 shows the 2-dimensional product of 7-node complete binary trees. The notion of "dimension" in product graphs will be made more precise in the next section, but for now it suffices to think of the r-dimensional product of graph G as a generalization from the r-dimensional grid, where each dimension is connected in the pattern of G.

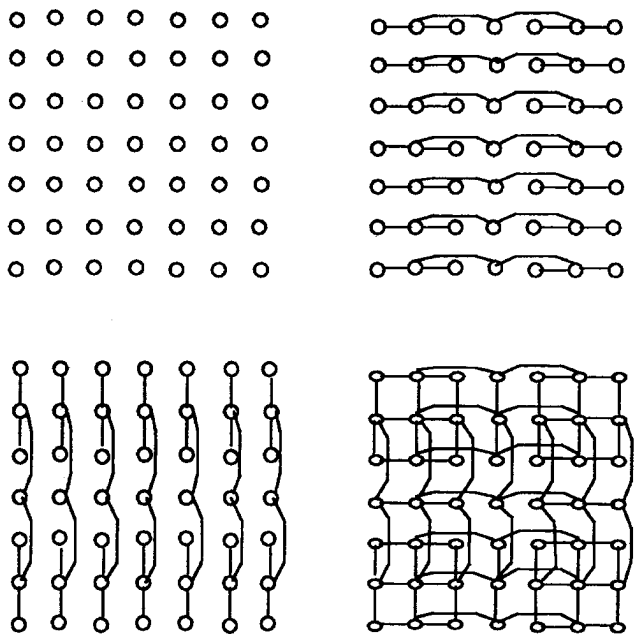


Fig. 1. Two dimensional product of complete binary trees. The grid points shown upper-left are connected in the binary tree pattern for each row and for each column. The final product network is shown lower-right.

Interconnection networks with small, fixed vertex degrees, and logarithmic diameters exist, including for instance binary trees, meshes of trees, shuffle-exchange, and de Bruijn networks, and these networks are good for those computations were the grid is not. However they are inefficient for other

computations where the grid interconnection is efficient. The power of hypercube is due to its ability to emulate all of these and other architectures efficiently [4], [13], [5], [7]. If we exclude the hypercube from consideration due to its large vertex degree, then we are faced with the challenge of designing a network which performs as well as the hypercube in these areas and which has a small and fixed vertex degree. As shown in this paper, product networks built from fixed degree networks can potentially serve as viable candidates for this purpose.

Some special cases of product networks have been studied by other researchers. For example, Rosenberg [17] showed that two dimensional products of de Bruijn networks can efficiently emulate grids, meshes of trees, de Bruijn networks and others. Section VI of our paper reinspects some of these resultsGanesan and Pradhan [12] studied the product graph obtained from crossing hypercubes with de Bruijn networks. The resulting network is analogous to the two dimensional grid whose connections in the first dimension are replaced for the hypercube connections, while the second dimension connections are replaced for that of de Bruijn graph. They showed that the resulting network has better embedding properties than the hypercube. Other well known product networks are grids where G is a linear array of N nodes, and hypercubes where $N = 2$. The generalized hypercube [3] can be also considered as the r-dimensional product of complete graphs.

This paper focuses on "homogeneous" product networks of r dimensions for $r \geq 2$; homogeneous in the sense that for every dimension, the pattern of interconnection is defined by the same graph G. The more general case of different interconnection patterns at different dimensions is certainly worthy of investigation. The choice of homogeneous products in this paper is not completely arbitrary however, since it allows the investigation of certain relationships between a "factor" network and its r-dimensional product versions more easily. That is, it is relatively easier to state and prove statements of the form "... if G has the property A, then the r-dimensional product of G has the corresponding property B... ." Not only this type of analyses give a clear picture of any improvement (or lack of it as the case may be) attained by product networks, but also certain facts of this type can be easily generalized for "heterogeneous" products.

Three cases are analyzed in this paper, including products of complete binary trees, shuffle-exchange, and de Bruijn networks. This selection is based on two reasons: First, these are already known to be powerful networks for parallel computation, and second, they have small and fixed vertex degrees. As shown in this paper, product networks based on these architectures are even more powerful computationally. On the negative side, the maximum vertex degree of product networks considered here grows faster (by a constant factor) than that of the hypercube or grid, with the increasing number of dimensions. However, these product networks can also grow without increasing the vertex degree. This latter property gives them an advantage over the hypercube which cannot grow without increasing the vertex degree.

After presenting the basic definitions and notations in the next section, the general properties of product networks are analyzed in Section III. These discussions are focused on those properties of product networks which are considered important for parallel computation; including the vertex degrees, partitionability, connectivity, diameter, bisection width, embedding properties, and routing.

The products of complete binary trees are considered in Section IV. Based on the results in Section III, it is shown that r-dimensional $N \times N \times \cdots \times N$ product of complete binary trees has diameter $D = 2r(Log\ (N + 1) - 1)$, maximum vertex degree $3r$, and bisection width at least $\Omega(N^{r-1})$. It can emulate the r-dimensional N^r-node torus with dilation 3 and congestion 2, and contains the r-dimensional mesh of N-node trees as a subgraph.

The products of shuffle-exchange graphs are considered in Section V. Again from the analyses of Section III, it is noted that the r-dimensional $N \times N \times \cdots \times N$ product of shuffle-exchange networks has diameter $D = r(2Log\ N - 1)$, maximum vertex degree $3r$, and bisection width $\Theta(N^r/LogN)$. It contains the r-dimensional N^r-node grid as a subgraph, and emulates the r-dimensional mesh of $(N - 1)$-node trees with dilation cost 2 and congestion cost 2. It is also shown that N^r-node pure shuffle-exchange graph can be embedded in the r-dimensional $N \times N \times \cdots \times N$ product of shuffle-exchange networks with dilation cost $2r$ (for $r > 1$) and congestion cost 2. This dilation can be considered as constant when r is fixed. Moreover, reverse embedding of the product of shuffle-exchange graphs on the pure shuffle-exchange graph is shown to require a logarithmic dilation cost, which suggests that the product version of shuffle-exchange network is computationally more powerful than the pure shuffle-exchange network itself.

Finally the products of de Bruijn networks are considered in Section VI. The r-dimensional $N \times N \times \cdots \times N$ product of de Bruijn networks has diameter $D = r(LogN)$, the maximum vertex degree $4r$, and bisection width is $\Theta(N^r/LogN)$. It *contains* the r-dimensional N^r-node torus as well as the r-dimensional mesh of $(N - 1)$-node trees as a subgraph. These are significant advantages over the other two product networks examined for a small increase in the vertex degree. It is further shown that N^r-node de Bruijn graph can be embedded in the r-dimensional $N \times N \times \cdots \times N$ product of de Bruijn networks with dilation cost r and congestion cost 4. Again, this dilation can be considered as constant when r is fixed. Moreover, reverse embedding of the product of de Bruijn graphs on the pure de Bruijn graph is shown to require a logarithmic dilation cost, which again suggests that the product version of de Bruijn network is computationally more powerful than the pure de Bruijn network itself.

The conclusion section discusses some of the open research areas.

II. Definitions and Notations

We mostly use undirected graphs to model interconnection networks, while occasionally taking advantage of directed edges to shorten certain proofs when no loss of generality occurs. It will often be important to indicate the number of vertices, so we use $G(N)$ to denote the N-node graph

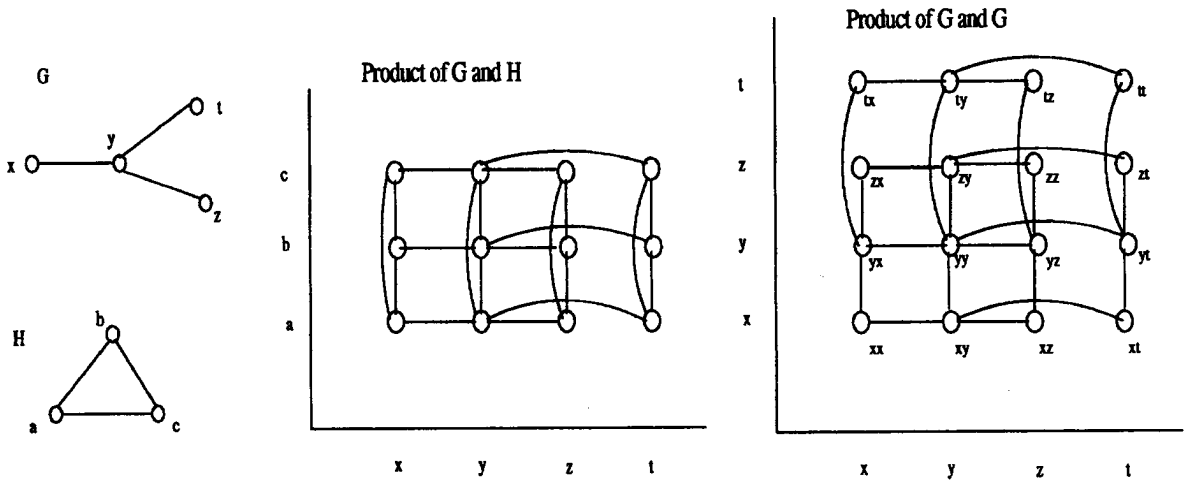


Fig. 2. Definition of product graphs.

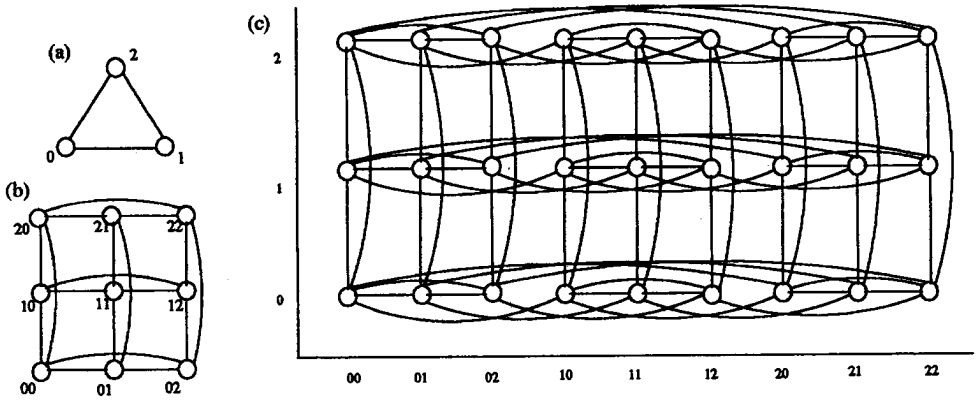


Fig. 3. Recursive construction of multi-dimensional product networks: (a) the factor graph, (b) two dimensional product, (c) three dimensional product.

G. The r-dimensional product of $G(N)$ is denoted $PG_r(N)$, with the subscript r representing the number of dimensions. These notations will be maintained for consistency throughout the paper, with a few exceptions when no confusion can arize due to the context of discussion.

In this paper, we let u, v, w denote the vertices of $G(N)$, and x, y, z denote the vertices of product graphs obtained from $G(N)$. Since $G(N)$ has N-vertices, the labels u, v, w take values $0, \cdots, (N-1)$. For the r-dimensional product graph $PG_r(N)$, the vertex labels x, y, z are strings of r symbols where each symbol is drawn from $\{0, \cdots, (N-1)\}$. For example, x is in the form $x = u_{r-1} \cdots u_i \cdots u_0$ where u_i is an N-valued symbol.

Additional notation will be introduced as needed. As a reminder to the reader, the definition of product graphs is provided first, and illustrated in Fig. 2. (This particular definition is frequently referred to as "cross product," as opposed to other product operations in the literature. We just use "product" to mean the cross product.)

DEFINITION 1. *The cross product of two graphs $G = (U, E)$ and $H = (V, F)$ is the graph $G \oplus H$ whose vertex set is $U \times V$, and edge set is defined as follows: Assume $\{u, u'\} \in U$ and $\{v, v'\} \in V$, then $(uv, u'v')$ is an edge in $G \oplus H$ if and only if either $u = u'$ and $(v, v') \in F$, or $v = v'$ and $(u, u') \in E$.*

From the symmetry in this definition, note that the product operator is commutative and associative. That is:

OBSERVATION 1. $G_1 \oplus G_2 = G_2 \oplus G_1$, and $G_1 \oplus (G_2 \oplus G_3) = (G_1 \oplus G_2) \oplus G_3$.

The formal definition of r-dimensional product graphs is given as follows:

DEFINITION 2. *Given a graph $G(N)$, the r-dimensional product, denoted $PG_r(N)$, is*

1) *a single vertex without any edges and no labels when $r = 0$,*
2) *$PG_r(N) = G(N) \oplus PG_{r-1}(N)$, when $r > 0$.*

At a more intuitive level, the construction of $PG_r(N)$ from $PG_{r-1}(N)$ can be described as follows: First, place the vertices of $PG_{r-1}(N)$ along a straight line as shown in Fig. 3. Then, draw N copies of $PG_{r-1}(N)$ such that the vertices with identical labels fall in the same column. Next extend the vertex labels, so that vertex label x becomes ux, for $u \in \{0 \cdots (N-1)\}$. Finally, connect the columns in the interconnection pattern of the

labeled graph $G(N)$, such that ux is connected to $u'x$ if and only if (u, u') is an edge in $G(N)$. From this, the edges of $PG_r(N)$ can be characterized as follows.

OBSERVATION 2. If x and y are in the form $x = u_{r-1} \cdots u_0$ and $y = v_{r-1} \cdots v_0$, where u_i, v_i are drawn from $\{0, \cdots, (N-1)\}$, then (x, y) is an edge in $PG_r(N)$ if and only if x and y differ in exactly one symbol position ℓ, and the differing symbols (u_ℓ, v_ℓ) define an edge in G.

This can be easily verified. Suppose two labels x and y differ in just one symbol position. From Observation 1, we can reorder the symbols in the labels so that the differing symbols become the leftmost symbol. Then the claim can be verified from Definition 2. If two or more symbols differ, one of them can be made the leftmost symbol and the claim can again be verified from Definition 2.

III. The General Properties of Product Networks

This section analyzes some computational properties which are common for all product graphs.

Among the first questions we ask about a new interconnection network is the number of nodes and links in it. We can easily observe that, if $G(N)$ has N vertices and E edges, then $PG_r(N)$ has N^r vertices and ErN^{r-1} edges. The statement about the number of vertices follows directly from Definition 2.[2] To compute the number of edges, observe from Fig. 3 that $PG_r(N)$ contains all the edges of N copies of $PG_{r-1}(N)$ plus the edges of N^{r-1} copies of $G(N)$ (there are N^{r-1} columns in Fig. 3). Thus we can write $f(r) = N f(r-1) + E N^{r-1}$ for the number of edges in $PG_r(N)$. The solution of this, with the initial condition $f(1) = E$, gives the desired result.

The vertex degree of a graph determines its connectivity and the number of parallel paths between an arbitrary pair of vertices in it (especially if the graph is regular). We can easily observe that if $G(N)$ has minimum vertex degree d_{min} and maximum vertex degree d_{max}, then $PG_r(N)$ has minimum vertex degree rd_{min} and maximum vertex degree rd_{max}. Note from Fig. 3 that each time we add a new dimension to the product network, we add at least d_{min} and at most d_{max} to the vertex degrees. Then, there is a vertex $x = u_{r-1} \cdots u_i \cdots u_0$ where each u_i corresponds to a vertex of $G(N)$ with degree d_{min}. This means x is a node with the minimum vertex degree of rd_{min}. Similarly, there is a vertex $y = v_{r-1} \cdots v_i \cdots v_0$ where each v_i corresponds to a vertex of $G(N)$ with degree d_{max}. Then y must have the maximum vertex degree rd_{max}.

From the construction of product networks, it can be easily shown that if every pair of vertices in $G(N)$ are connected by at least κ vertex-disjoint paths, then every pair of vertices in $PG_r(N)$ are connected by at least $r\kappa$ vertex-disjoint paths. A network with more parallel paths between arbitrary pairs of nodes has higher communication bandwidth and better fault tolerance capabilities.

2. Actually homogeneous product networks do not always have to contain N^r nodes. If $G(N)$ can be built with different N values, then we can build the product version by combining different size networks at different dimensions and fine tune the overall size to match the desired size. For simplicity in discussions, we assume in this paper that the value of N is same for every dimension.

The ability to recursively partition a graph into distinct copies of its smaller versions is another important property, since it allows assigning the parts of a recursive computation to different subnetworks, or shows a way to share the system between many users. Product graphs contain a variety of subgraphs which are isomorphic copies of product graphs of lower dimensions. Let $PG_r^{-i}(N)$ denote the subgraph of $PG_r(N)$ induced by removing the symbol at position i. This corresponds to erasing the connections at dimension i. Then, for $r > 0$, $PG_r^{-i}(N)$ is isomorphic to N disjoint copies of $PG_{r-1}(N)$ for any $i \in \{0, \cdots (r-1)\}$. For $i = r-1$ (i.e., the leftmost symbol index) the partionability is immediate from Definition 2 and Fig. 3. For arbitrary i, we can use Observation 1 to reorder the symbols in the address labels so that the ith symbol becomes the leftmost symbol, and then refer to Definition 2. This can be applied recursively and any number of symbols can be removed from the vertex labels to obtain product graphs of smaller dimensions.

The diameter of a network is another important property. Several papers were devoted to developing networks with small diameter and small vertex degree, and bounds have been derived on diameter as a function of vertex degree [14]. In general, computation of exact diameter for a given graph may be difficult, but for homogeneous product graphs we are able to state simple rules to calculate the diameter. The next two results (or their restricted versions) have been discovered in many papers independently [1], [8], [12], [17].

THEOREM 1. *If $G(N)$ has diameter D then $PG_r(N)$ has diameter rD.*

THEOREM 2. *If $G(N)$ is a self-routing network then $PG_r(N)$ is also a self-routing network.*

We say that a network is "self-routing" if messages can be delivered to their destinations through shortest paths without an external controller. The basic idea in Theorem 2 is to apply the routing algorithm of $G(N)$ in each dimension of $PG_r(N)$ where the source and destination addresses differ. The basic idea in Theorem 1 is to observe that there exist pairs of nodes in $PG_r(N)$ which differ in every symbol position. Moreover, each differing symbol pair may correspond to a distance as much as the diameter of $G(N)$. Finding such a pair yields both a lower bound and an upper bound for the diameter of the product graph.

The embedding results of this research are among the most important results since they show a way of emulating one network by another. In the context of product networks, the utility of embedding results is further emphasized by the fact that many of the existing popular architectures can be modeled as product networks. An embedding of a "guest" graph G in a "host" graph H is a mapping of the vertices of G into the vertices of H and the edges of G into paths in H. The main cost measures used in embedding efficiency are:

- *Load* of the embedding is the maximum number of vertices of G mapped to any vertex of H.
- *Dilation* of an embedding is the maximum path length in H representing an edge of G.

- *Congestion* of an embedding is the maximum number of paths (that correspond to the edges of G) that share any edge of H.

It was shown in [15] that if G can be embedded in H with load l, dilation d, and congestion c, H can emulate t steps of a computation running on G in $O(l + d + c)t$ steps. If the values l, d, and c are constant, the slowdown introduced by this emulation is also constant. We will make frequent use of the following result. Let G and H be labeled graphs. Then:

THEOREM 3. *$PG_r(N_G)$ is a subgraph of $PH_r(N_H)$ if and only if $G(N_G)$ is a subgraph of $H(N_H)$.*

PROOF. The sufficient condition has been shown before (Lemma 3.3 in [16]). Therefore, we only focus on the necessary condition. If $G(N_G)$ is not a subgraph of $H(N_H)$, then there must be at least one edge (u, v) in $G(N_G)$ which cannot be mapped to any edge in $H(N_H)$. Since $PG_r(N_G) = G(N_G) \oplus PG_{r-1}(N_G)$, we can write an edge of $PG_r(N_G)$ as (ux, vx) where x is a vertex in $PG_{r-1}(N_G)$. Similarly, $PH_r(N_H) = H(N_H) \oplus PH_{r-1}(N_H)$, and the edge (ux, vx) cannot exist in $PH_r(N_H)$ since (u, v) is not an edge in $H(N_H)$. □

This theorem and its extensions have many significant implications. In particular, the next two results will be used frequently in the following sections.

COROLLARY 1. *If $G(N_G)$ can be embedded in $H(N_H)$ with dilation d, then $PG_r(N_G)$ can be embedded in $PH_r(N_H)$ with dilation d.*

PROOF. Modify $G(N_G)$ to obtain $G'(N_{G'})$, such that whenever an edge e of $G(N_G)$ is mapped to a path in $H(N_H)$, replace it for the path it is mapped to (see Fig. 4). Since $G'(N_{G'})$ obtained this way is a subgraph of $H(N_H)$, apply the above theorem. □

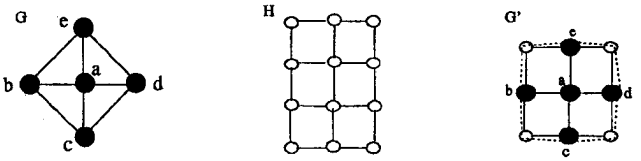


Fig. 4. Obtaining G' from G.

COROLLARY 2. *If $G(N_G)$ can be embedded in $H(N_H)$ with congestion c, then $PG_r(N_G)$ can be embedded in $PH_r(N_H)$ with congestion c.*

PROOF. An embedding of $G(N_G)$ in $H(N_H)$ with congestion cost c directly induces the claimed embedding for $G(N_G)$ in $H(N_H)$. □

An interesting result of these corollaries is that every product graph of r dimensions can emulate the r-dimensional torus with dilation cost 3 and congestion cost 2. This follows from a theorem due to Leighton [16] which states that the N-node cycle can be embedded in any N-node connected graph G with dilation cost 3 and congestion cost 2. Using this fact in the above corollaries leads to the claimed fact.

The following is useful when proving embedding results. It has been used quite frequently, and often implicitly by many researchers, and this makes it difficult to attribute it to a single researcher.

PROPOSITION 1. *Let G' be a subgraph of G.*

1) *If G can be embedded in H with dilation cost d, then G' can be embedded in H with dilation cost at most d.*
2) *If every embedding of G' in H requires a dilation cost d, then every embedding of G in H requires a dilation cost at least d.*
3) *If every embedding of H in G requires a dilation cost d, then every embedding of H in G' requires a dilation cost at least d.*

PROOF. The first part of the proposition is easily seen since an embedding of G in H induces an embedding for every subgraph of G in H. For the second part, assume for the sake of argument that G' requires a dilation d but G can be embedded in H with dilation d_1, where $d_1 < d$. If this were the case, we could embed G' in G with unit dilation and then embed G in H. From part 1 of the proposition this induces an embedding for G' in H with dilation at most d_1, which shows that d could not have been the best dilation obtainable for embedding G' in H. The third item is easy to see since G' only contains a subset of the edges of G. □

We note that these statements are equally true for the congestion of embedding also. That is, we could replace the word "dilation" for "congestion" and the statements would continue to hold.

The bisection width of a network determines its bandwidth, and has important implications about the VLSI layout complexity bounds [19]. We first give a definition which will be used in the statement and proof of the following theorem.

DEFINITION 3. *We say that the* "maximal congestion" *of a connected graph $G(N)$ is C if after mapping the vertices of the N-node directed complete graph onto the vertices of $G(N)$ in a one-to-one manner there is a mapping of the edges of the complete graph into paths in $G(N)$ such that no edge of $G(N)$ has congestion more than C.*

Note that the maximal congestion is an intrinsic parameter of a graph just like the chromatic number, crossing number, etc. are intrinsic parameters of a graph.

THEOREM 4. *If the maximal congestion of $G(N)$ is C then $PG_r(N)$ has bisection width at least $\frac{N^{r+1}}{2C}$. If N is even and $G(N)$ has bisection width $B = \frac{N^2}{2C}$, this bisection width is exact.*

PROOF. First we show that when N is even and $G(N)$ has bisection width $B = \frac{N^2}{2C}$ the claimed value is an upper bound. Then we show that for all values of N the claimed value is also a lower bound. □

Upper bound: Consider *any* partition of $G(N)$ into two subgraphs each containing half the number of nodes. We obtain a partition for $PG_r(N)$ by applying this partition to all of the $G(N)$ graphs in a given dimension. If N is even, each part of $PG_r(N)$ obtained this way contains $N^r/2$ nodes. Let (u, v) be one of those

edges cut by the partition applied to $G(N)$, and let B denote the bisection width of $G(N)$. Recall that $PG_r(N) = G(N) \oplus PG_{r-1}(N)$, and if (u, v) is an edge in $G(N)$, then (ux, vx) is an edge in $PG_r(N)$, where $x = u_{r-2} \cdots u_0$ is a vertex in $PG_{r-1}(N)$. Since x can take N^{r-1} possible values, and removal of B edges completely disconnects $G(N)$, the corresponding total number of edges removed from $PG_r(N)$ is BN^{r-1}. If B is the *minimum* bisection width for $G(N)$, then $U = BN^{r-1}$ is a good upper bound for the bisection width of $PG_r(N)$. If $B = \frac{N^2}{2C}$ then $U = \frac{N^{r+1}}{2C}$ is the claimed upper bound when N is even. We give an example.

EXAMPLE 1. Consider the special case where the factor graph is $K(N)$, the directed complete graph of N nodes, where N is even. Since $K(N)$ has bisection width $N^2/2$, the upper bound U_{PK} for the bisection width of r-dimensional product $PK_r(N)$ is $U_{PK} = BN^{r-1} = N^{r+1}/2$.

Lower bound: We use a proof technique extended from Theorem 1.21 in [16]. First of all, we know that the bisection width of N-node directed complete graph is $N^2/2$. After mapping the edges of the directed complete graph on the available paths in $G(N)$, assume the maximal congestion for any edge of $G(N)$ is C. Then, if *every* edge cut in the bisection of $G(N)$ *represents* exactly C edges of the directed complete graph then the bisection width of $G(N)$ is $L = \frac{N^2}{2C}$. This is because a bisection for $G(N)$ also induces a bisection for the directed complete graph. Note however that, L is a lower bound because some edges cut in the bisection of $G(N)$ may represent less than C edges of the directed complete graph, depending on how the edges of the complete graph are routed on $G(N)$.

Thus, if we know C (as assumed in the statement of the theorem), we can compute a lower bound as $L \geq \frac{N^2}{2C}$.

EXAMPLE 2. To compute a lower bound L_{PK} on the bisection width of $PK_r(N)$ by this method, we first map the nodes of N^r-node directed complete graph, $K(N^r)$, onto the nodes of $PK_r(N)$ one to one. We then map the edges of $K(N^r)$ to shortest paths in $PK_r(N)$. Consider an edge (x, z) of $K(N^r)$ mapped to a path in $PK_r(N)$. Let (x, y) be the first edge of the path from x to z. If y differs from x in dimension ℓ, then z must also differ from x in dimension ℓ, and the edge (x, y) must be used for all possible values of z. Since z can take at most N^{r-1} different values (including y itself), the congestion of the edge (x, y) is at most $C_{PK} = N^{r-1}$. Then the lower bound is $L_{PK} = \frac{N^{2r}/2}{C_{PK}}$, where $N^{2r}/2$ is the bisection width of $K(N^r)$. This yields $L_{PK} = N^{r+1}/2$, which is equal to U_{PK} above.

These two examples establish that the exact bisection width of $PK_r(N)$ is $N^{r+1}/2$. We now use this to prove the theorem. Above, we showed that each edge of $PK_r(N)$ represents exactly N^{r-1} edges of $K(N^r)$. From the statement of the theorem, we know that each edge of $G(N)$ represents at most C edges of $K(N)$. Therefore each edge of $PG_r(N)$ represents at most CN^{r-1} edges of $K(N^r)$. Since the bisection width of $K(N^r)$ is $N^{2r}/2$, a lower bound on the bisection width of $PG_r(N)$ is $\frac{N^{2r}/2}{CN^{r-1}} = N^{r+1}/2C$ as claimed.

Now consider the case for odd values of N. As in Example 2, we can compute the maximum congestion induced from mapping the $K(N^r)$ graph on $PK_r(N)$ as $C_{PK} = N^{r-1}$. Again the theorem assumes that the congestion of mapping $K(N)$ on $G(N)$ is at most C. Then, each edge of $PG_r(N)$ represents at most CN^{r-1} edges of $K(N^r)$. Since the bisection width of $K(N^r)$ is $\frac{N^{2r}-1}{2}$ when N is odd, the bisection width of $PG_r(N)$ is at least $\frac{N^{2r}-1}{2CN^{r-1}}$. This differs from the case of even N by $\frac{1}{2CN^{r-1}}$ which diminishes as N or r grows.

Both Theorems 3 and 4 are very significant since they simplify the analyses of product networks. From Theorem 3, to understand if the product of one graph is a subgraph of another, we just need to focus on the corresponding factor graphs which are much simpler than their products. From Theorem 4, to compute a lower bound on bisection width of the r-dimensional product network, we just need to compute the maximal congestion for the factor graph. This can be simply done if the factor graph has a routing algorithm: from each node, send one packet to every other node, and count the maximum number of packets that traces any edge. This gives the maximal congestion. Alternatively, if we have an exact bisection width value for the factor graph we can use it to obtain a lower bound for the product graph.

IV. Products of Complete Binary Trees

A. Basic Properties

A complete binary tree of $N = 2^h - 1$ nodes, denoted $T(N)$, has a minimum vertex degree of 1 and a maximum vertex degree of 3. Any two nodes are connected by exactly one path, the diameter is $2(Log(N + 1) - 1)$, and the bisection width is 1.

A two dimensional product of complete binary trees is shown in Fig. 1. The r-dimensional product of binary trees is obtained as $T(N) \oplus PT_{r-1}(N)$. The following properties of $PT_r(N)$ are immediately observed from the results in the previous section:

1) $PT_r(N)$ has the minimum vertex degree r, and the maximum vertex degree $3r$.
2) $PT_r(N)$ contains N^r vertices and $r(N-1)N^{r-1}$ edges.
3) $PT_r(N)$ contains N copies of the $PT_{r-1}(N)$ subgraph.
4) Every pair of vertices in $PT_r(N)$ are connected by at least r vertex-disjoint paths, and at most $3r$ vertex-disjoint paths.
5) Diameter of $PT_r(N)$ is $2r(Log(N + 1) - 1)$ and bisection width is at least $\Omega(N^{r-1})$.
6) There exists a shortest path routing algorithm for the $PT_r(N)$ graph as a simple extension of the binary tree shortest path routing algorithm.

B. Embedding Properties

Despite their simple structures, products of binary trees have very interesting embedding properties. For instance, while tori and meshes of trees are powerful architectures, they have different strengths and weaknesses. It is shown in this section that the product of binary trees can emulate both of these architectures very efficiently. It is further shown that $PT_r(N)$ can emulate a

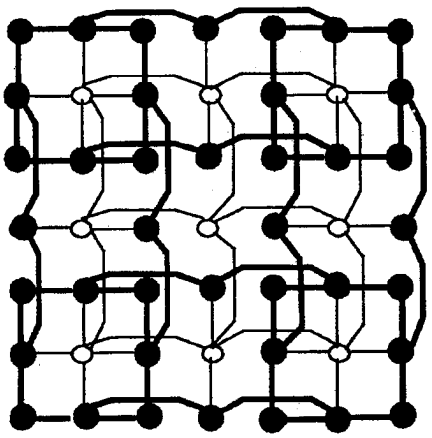


Fig. 5. Embedding meshes of trees in products of complete binary trees.

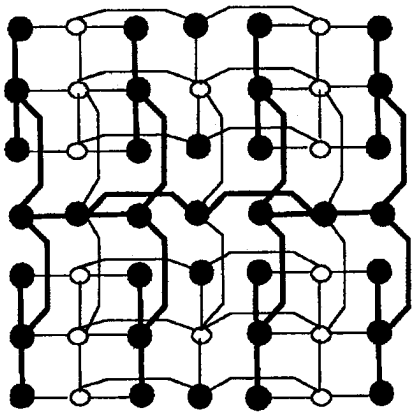


Fig. 6. Embedding of the complete binary tree in the two dimensional product of binary trees.

comparable size complete binary tree efficiently, while the reverse emulation of the $PT_r(N)$ architecture by a complete binary tree requires logarithmic dilation cost.

THEOREM 5. *r-dimensional $N \times \cdots \times N$ torus can be embedded in $PT_r(N)$ with dilation cost* 3 *and congestion cost* 2.

PROOF. Due to Corollaries 1 and 2, it suffices to show that N-node cycle can be embedded in N-node complete binary tree with dilation cost 3 and congestion cost 2. This follows from a theorem due to Leighton [16] which states that the N-node cycle can be embedded in any N-node connected graph with dilation cost 3 and congestion cost 2. □

The $PT_r(N)$ graph contains not just the mesh of trees, but a hierarchy of meshes of trees as shown next.

THEOREM 6. *For all $i = 1, \cdots, Log(N+1) - 1$, $PT_r(N)$ contains the mesh of $(N+1)/2^i$-leaf trees.*

PROOF. Fig. 5 shows the two dimensional meshes of trees contained in $PT_2(7)$. Note that in this figure there are two meshes of trees contained; one with $(N+1)/2 = 4$ leaves for each tree (shown in dark nodes), and one with $(N+1)/4 = 2$ leaves for each tree (shown in empty nodes). In general, the largest mesh of trees contained in $PT_r(N)$ is obtained as follows: Start with the $PT_2(N)$ graph, and enumerate the row trees $0, \cdots, (N-1)$. For those trees with even numbers, color the leaves in red, and internal nodes in blue. Do not color the odd numbered trees. For the second dimension, color the internal nodes of a tree in blue, if and only if it has red leaves. Note that this coloring scheme is consistent with the construction of two dimensional mesh of trees; that is, all the colored nodes, red or blue, are contained in the two dimensional mesh of trees. For induction, assume that $(r-1)$-dimensional mesh of trees has been already colored in the $PT_{r-1}(N)$ graph. The coloring rule for the r-dimensional mesh of trees, is same. That is, when going from $PT_{r-1}(N)$ to $PT_r(N)$, color the internal nodes of a highest dimension tree in blue, if and only if it has red leaves. The set of vertices colored red or blue gives the largest r-dimensional mesh of trees contained in $PT_r(N)$.

Once the largest mesh of trees is colored, successively smaller meshes of trees are then obtained by removing all the colored vertices, and coloring the remaining vertices by the same strategy. □

The next two results show that $PT_r(N)$ is strictly more powerful than the corresponding size complete binary tree.

THEOREM 7. *For $r > 1$, the complete binary tree of $r(h-1)+1$ levels is a subgraph of $PT_r(N)$, where $N = 2^h - 1$.*

PROOF. For $r = 2$, the embedding of 5-level complete binary tree in $PT_2(7)$ is shown in Fig. 6. Note in particular that the tree in the middle row constitutes the highest three levels of the tree. The leaves of this row tree correspond to the roots of column trees. This pattern can be recursively repeated for larger values of N in two dimensions. Assuming that the claim is true for $PT_{r-1}(N)$, the embedding proof for r dimensions follows from the recursive construction of $PT_r(N)$. □

Note that for $r = 2$, the tree embedded by the above method is the largest tree possible. The next result shows that complete binary tree cannot emulate its comparable size product network with less than logarithmic dilation.

THEOREM 8. *Any embedding of $PT_r(N)$ in the large enough complete binary tree requires dilation cost $\Omega(Log(rLogLogN))$.*

PROOF. Referring to Proposition 1, we show that $PT_r(N)$ contains a subgraph G_1, and there exists a graph G_2 which contains the complete binary tree as a subgraph, such that embedding of G_1 in G_2 requires the claimed amount of dilation.

G_1 is the r-dimensional grid. Since N-node complete binary tree contains a path of length $L = 2(Log(N+1) - 1)$, $PT_r(N)$ contains an $M = L^r$ node grid. We can select G_2 as the de Bruijn graph since it contains the complete binary tree as a subgraph. It is shown in [2] that any embedding of r-dimensional (for $r > 1$) M node grid in de Bruijn graph requires a dilation of at least $\Omega(LogLogM)$. Using $M = L^r$ gives the claimed result. □

V. PRODUCTS OF SHUFFLE-EXCHANGE GRAPHS

The N-node shuffle-exchange graph, denoted $S(N)$, contains $N = 2^k$ nodes, labeled $0, \cdots, 2^k - 1$ (in binary), and $3 \times 2^{k-1}$ edges connected as follows: (u, v) is a directed edge if and

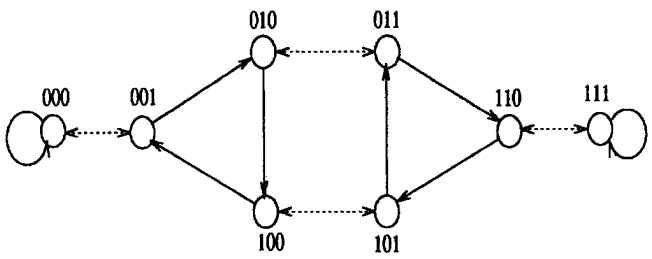


Fig. 7. The 8-node shuffle-exchange graph with shuffle edges shown in solid lines and exchange edges shown in dotted lines.

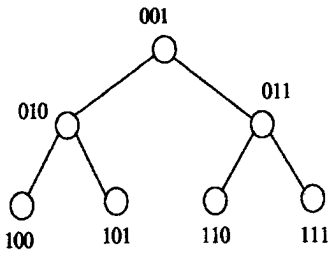


Fig. 8. The level order labeling of complete binary tree.

only if either a) u and v differ in the rightmost bit only, called the "exchange" edges, or, b) v can be obtained from u by a cyclic left shift, called the "shuffle" edges. An 8-node shuffle-exchange graph is shown in Fig. 7. In the shuffle-exchange graph, every vertex has a degree of 3. Any two nodes are connected by at least one path, although many pairs of vertices are connected by up to three vertex disjoint paths. The diameter is $2LogN - 1$ and bisection width is $\Theta(N/LogN)$.

The r-dimensional product of shuffle-exchange graph, denoted $PS_r(N)$, is obtained as $PS_r(N) = S(N) \oplus PS_{r-1}(N)$. The following properties of $PS_r(N)$ are immediately observed from the results in Section III:

1) Every node in $PS_r(N)$ has vertex degree $3r$ (although some edges are self loops).
2) $PS_r(N)$ contains N^r vertices and $r(3N/2)N^{r-1}$ edges.
3) $PS_r(N)$ contains N copies of the $PS_{r-1}(N)$ subgraph.
4) Any pair of vertices in $PS_r(N)$ are connected by at least r vertex-disjoint paths, and at most $3r$ vertex-disjoint paths.
5) Diameter of $PS_r(N)$ is $r(2LogN - 1)$ and bisection width is $\Theta(N^r/LogN)$.
6) There exists a routing algorithm for the $PS_r(N)$ graph as a simple extension of the shuffle-exchange routing algorithm.

A. Embedding Properties

It is first shown that products of binary trees can be embedded in the products of shuffle-exchange graphs with dilation cost 2 and congestion cost 2. While this result carries all the embedding properties of $PT_r(N)$ to the $PS_r(N)$ graph, it may be better to find direct embeddings for some cases. For instance, r-dimensional grids are subgraphs of the $PS_r(N)$. Next, it is shown that the N^r-node shuffle-exchange graph can be embedded in the $PS_r(N)$ graph with dilation cost $2r$ and congestion cost 2. For an implementation with a fixed number of dimensions, this embedding can be considered as constant dilation, particularly because N can be independent of r. Moreover, it is shown that $PS_r(N)$ cannot be embedded in the N^r-node shuffle-exchange graph with less than logarithmic dilation cost. This makes the product network more powerful than the shuffle-exchange network itself.

THEOREM 9. *$PT_r(N - 1)$ can be embedded in $PS_r(N)$ with dilation cost 2 and congestion cost 2.*

PROOF. Due to Corollaries 1 and 2, it suffices to show that the $(N - 1)$-node binary tree can be embedded in the N-node shuffle-exchange graph with dilation cost 2 and congestion cost 2. As it is well known, the level order labeling of the $(N - 1)$-node complete binary tree as shown in Fig. 8 induces the desired embedding. The root is assigned the label 1, and successively lower levels are assigned the remaining labels left-to-right. □

The following results are now immediately observed:

COROLLARY 3. *As in Theorem 6, a hierarchy of meshes of trees can be embedded in $PS_r(N)$ with dilation cost 2 and congestion cost 2.*

COROLLARY 4. *The r-dimensional N^r-node grid is a subgraph of $PS_r(N)$.*

PROOF. It was shown in [9] that the shuffle-exchange network contains a hamiltonian path. Hence the result follows from Theorem 3. □

The next two results consider the embedding of shuffle-exchange graph in its product version, and the reverse embedding of product network in the pure shuffle-exchange graph.

THEOREM 10. *For $r > 1$, the N^r-node shuffle-exchange graph can be embedded in $PS_r(N)$ with dilation cost $2r$ and congestion cost 2.*

PROOF. First consider the case for $r = 2$. Both $S(N^2)$ and $PS_2(N)$ are labeled by $2LogN$-bit strings. For the product graph, the rightmost $LogN$ bits determine the "row address," while the leftmost $LogN$ bits determine the "column address." We show that whenever (u, v) is an exchange edge in $S(N^2)$, it is also an exchange edge in $PS_2(N)$. Alternatively, whenever (u, v) is a shuffle edge in $S(N^2)$, there is a path of length at most 4 from u to v in $PS_2(N)$.

Consider the vertex:

$$u = u_{2n-1}u_{2n-2}\cdots u_{n+1}u_n \,|\, u_{n-1}u_{n-2}\cdots u_1u_0$$

Here "|" separates the left hand half of the label from the right hand half. It suffices to focus on the outgoing edges only. If v is the exchange neighbor of u in $S(N^2)$, then

$$v = u_{2n-1}u_{2n-2}\cdots u_{n+1}u_n \,|\, u_{n-1}u_{n-2}\cdots u_1\bar{u}_0$$

In the $PS_2(N)$ graph, u has an exchange neighbor w in its row, whose address is obtained by complementing the rightmost bit of the address. That is $w = v$. In fact, it is true for arbitrary r that whenever (u, v) is an exchange edge of the N^r-node shuffle-exchange graph, it is also an exchange edge in the $PS_r(N)$ graph. Therefore, the rest of this proof only needs to consider the shuffle edges.

Now suppose (u, v) is a shuffle edge in $S(N^2)$. If u is as above, v must be:

$$v = u_{2n-2} \cdots u_{n+1} u_n \, u_{n-1} \,|\, u_{n-2} \cdots u_1 u_0 u_{2n-1}.$$

For the $PS_2(N)$ graph, the row neighbors of u are

$$w^{e,r} = u_{2n-1} u_{2n-2} \cdots u_{n+1} u_n \,|\, u_{n-1} u_{n-2} \cdots u_1 \overline{u}_0$$

and

$$w^{s,r} = u_{2n-1} u_{2n-2} \cdots u_{n+1} u_n \,|\, u_{n-2} \cdots u_1 u_0 \, u_{n-1}$$

where the superscripts "e,s,r" stand for "exchange," "shuffle," and "row," respectively. The column neighbors of u, indicated by the superscript "c," are

$$w^{e,c} = u_{2n-1} u_{2n-2} \cdots u_{n+1} \overline{u}_n \,|\, u_{n-1} u_{n-2} \cdots u_1 u_0$$

and

$$w^{s,c} = u_{2n-2} \cdots u_{n+1} u_n \, u_{2n-1} \,|\, u_{n-1} u_{n-2} \cdots u_1 u_0$$

In the following discussion, subscripts ℓ and r are used to denote the left-hand half of a label and the right-hand half of a label. For example, $w_\ell^{s,c}$ denotes the left-hand half of the vertex $w^{s,c}$ above. There are two cases to consider:

Case 1. $u_{2n-1} = u_{n-1}$. In this case the reader can easily verify that $v = w_\ell^{s,c} \,|\, w_r^{s,r}$. This means that one can go from u to v in $PS_2(N)$ in two steps; by moving to the shuffle neighbor of u in the column and then the shuffle neighbor in the row. Alternatively, one can move to the shuffle neighbor in the row first, and then in the column.

Case 2. $u_{2n-1} \neq u_{n-1}$. Then, given u and v as above, the left-hand half and the right-hand half of v can be computed as:

$$v_\ell = w_\ell^{s,c} + w_\ell^{e,c}$$

and

$$v_r = w_r^{s,r} + w_r^{e,r}$$

where the "+" sign denotes sequencing of the two moves. That is, $w_r^{s,c} + w_r^{e,c}$ denotes moving to the shuffle neighbor in the column, followed by moving to the exchange neighbor in the column. Since $v = v_\ell \,|\, v_r$, a sequence of four moves yields the desired vertex label.

To extend these arguments for $r > 2$, since $PS_r(N) = S(N) \oplus PS_{r-1}(N)$, a vertex of $PS_r(N)$ can be written as $u = u_{rn-1} u_{rn-2} \cdots u_{(r-1)n} \,|\, S'$, where S' is a vertex in $PS_{r-1}(N)$. For the discussion below, only the leftmost bit of S' is relevant, so we can write $S' = sS$. That is;

$$u = u_{rn-1} u_{rn-2} \cdots u_{(r-1)n} \,|\, sS.$$

In the N^r-node shuffle-exchange graph, the shuffle neighbor is

$$v = u_{rn-2} \cdots u_{(r-1)n} s \,|\, S u_{rn-1}$$

For the product network, u has a shuffle neighbor x^s, where

$$x^s = u_{rn-2} \cdots u_{(r-1)n} u_{rn-1} \,|\, sS$$

which in turn has an exchange neighbor x^e, where

$$x^e = u_{rn-2} \cdots u_{(r-1)n} \overline{u}_{rn-1} \,|\, sS$$

Let x_ℓ denote the leftmost $LogN$-bit substring of x (i.e. the part to the left of "$|$" above). Then, observe that

$$v_\ell = \begin{cases} x_\ell^s & \text{if } u_{rn-1} = s, \\ x_\ell^e & \text{otherwise} \end{cases}$$

That is, x^e is at a distance of two from u, and going from u to x^e corrects just the leftmost $LogN$ bits of the address towards v. Since the next set of $LogN$ bits can be corrected by the same method as above, $2(r-1)$ additional steps are needed to reach v. This completes the proof that dilation of embedding is $2r$.

To study the congestion, consider two vertices u, u' of $S(N^r)$. For the discussion below, we focus on the leftmost $LogN$ bits and the rightmost $LogN$ bits of these vertices and use $S' = sS$ to denote a vertex in $PS_{r-2}(N)$. Let

$$u = u_{rn-1} u_{rn-2} \cdots u_{(r-1)n} \,|\, sS \,|\, u_{n-1} u_{n-2} \cdots u_0$$

and

$$u' = \overline{u}_{rn-1} u_{rn-2} \cdots u_{(r-1)n} \,|\, sS \,|\, u_{n-1} u_{n-2} \cdots u_0$$

whose respective shuffle neighbors are

$$v = u_{rn-2} \cdots u_{(r-1)n} s \,|\, S u_{n-1} \,|\, u_{n-2} \cdots u_0 u_{rn-1}$$

and

$$v' = u_{rn-2} \cdots u_{(r-1)n} s \,|\, S u_{n-1} \,|\, u_{n-2} \cdots u_0 \overline{u}_{rn-1}$$

For the edges (u, v) and (u', v'), the corresponding paths in $PS_r(N)$ meet at the node

$$u_{rn-2} \cdots u_{(r-1)n} s \,|\, sS \,|\, u_{n-1} u_{n-2} \cdots u_0$$

after the leftmost $LogN$ bits have been corrected (by traversing one or two edges, depending on whether $u_{rn-1} = s$). From there, both paths share the same set of edges until the node

$$u_{rn-2} \cdots u_{(r-1)n} s \,|\, S u_{n-1} \,|\, u_{n-2} \cdots u_0 u_{n-1}$$

is reached. Since no other paths contain these edges the congestion thus far is 2.

If $u_{n-1} = u_{rn-1}$, then the vertex reached is v and the edge (u, v) has been completely mapped. However, the path from u' to v' still needs to traverse an exchange edge to invert its rightmost bit. The path only shares this edge with the exchange edge (v, v') in $S(N^r)$, and the congestion of the edge is 2.

If $u_{n-1} = \overline{u}_{rn-1}$, then the vertex reached is v', and the path from u to v still needs to traverse an exchange edge to invert its rightmost bit as before. Thus the congestion is again 2 and the proof is complete. □

THEOREM 11. *Any embedding of $PS_r(N)$ in $S(N^r)$ requires dilation $\Omega(Log(rLogN))$.*

Proof is deferred until after Theorem 15.

VI. PRODUCTS OF DE BRUIJN NETWORKS

The de Bruijn graph, denoted $D(N)$, contains $N = 2^k$ nodes, labeled 0, $\cdots$, $2^k - 1$ (in binary), and $2N$ edges connected as follows: (u, v) and (u, w) are directed edges if and only if v can be obtained from u by a cyclic left shift, and w differs from v in the rightmost bit only. An 8-node de Bruijn graph is shown in Fig. 9. Observe that whenever (u, v) is a shuffle-edge in $S(N)$, it is also an edge in $D(N)$. Additionally, whenever (u, v, w)

is a path in $S(N)$ such that (u, v) is a shuffle edge and (v, w) is an exchange edge, (u, w) is an edge in $D(N)$. Every vertex has degree 4. Every pair of nodes are connected by at least two vertex disjoint paths (since de Bruijn graph contains a hamiltonian cycle), although most pairs of vertices are connected by up to four vertex disjoint paths. The diameter is $LogN$ and bisection width is $\Theta(N/LogN)$.

The r-dimensional product of de Bruijn network, denoted $PD_r(N)$, is defined as $PD_r(N) = D(N) \oplus PD_{r-1}(N)$. The following properties of $PD_r(N)$ are immediately observed from the results in Section III:

1) $PD_r(N)$ has the vertex degree $4r$.
2) $PD_r(N)$ contains N^r vertices and $2rN^r$ edges.
3) $PD_r(N)$ contains N copies of the $PD_{r-1}(N)$ subgraph.
4) Any pair of vertices in $PD_r(N)$ are connected by at least $2r$ vertex-disjoint paths, and at most $4r$ vertex-disjoint paths.
5) Diameter of $PD_r(N)$ is $rLogN$ and bisection width is $\Theta(N^r/LogN)$.
6) There exists a routing algorithm for the $PD_r(N)$ graph based on the de Bruijn routing algorithm.

Comparing to the product networks in the previous subsections, the vertex degree increases by 25%, while $PD_r(N)$ has better properties in other respects. Diameter reduces by 50%, and the minimum number of parallel paths between an arbitrary pair of vertices doubles. It also has better embedding properties as will be shown below.

A. Embedding Properties

It is well known that shuffle-exchange and de Bruijn networks are computationally equivalent. That is, every computation which can be performed on one of them, can be also performed on the other with constant slowdown. It is therefore reasonable to expect that their product versions would also be computationally equivalent. This result is formally stated by the following lemma.

LEMMA 1. *For every edge (x, y) in $PS_r(N)$, there is a path of length 2 or less in $PD_r(N)$. Conversely, for every edge (x, y) in $PD_r(N)$, there is a path of length 2 or less in $PS_r(N)$.*

PROOF. If (x, y) is an edge of $PS_r(N)$, where $x = u_{r-1} \cdots u_0$, and $y = v_{r-1} \cdots v_0$, by Observation 2, there must be just one differing symbol position i, where (u_i, v_i) is an edge in $S(N)$. Since this edge corresponds to a path of length not more than 2 in $D(N)$, it follows that (x, y) corresponds to a path of length not more than 2 in $PD_r(N)$. The converse case can be shown similarly. □

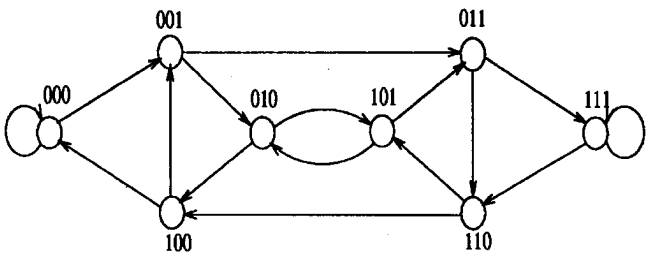


Fig. 9. The 8-node de Bruijn graph.

THEOREM 12. *For all $k \le N$, r-dimensional $k \times \cdots \times k$ torus is a subgraph of $PD_r(N)$.*

PROOF. Due to Theorem 3, it suffices to note that the de Bruijn network is pancyclic [18]; that is, for every value of $k \le N$, $D(N)$ contains a cycle of length k. □

This is a straight extension of corollary 3.4 in [17]. The next result, however, is more interesting:

COROLLARY 5. *$PD_r(N)$ contains k-dimensional $2^{j_1} \times 2^{j_2} \times \cdots \times 2^{j_k}$ grid, where $j_1 + j_2 + \cdots j_k = r$, as a subgraph.*

PROOF. Follows from the fact that $PD_r(N) = PD_{j_1}(N) \oplus PD_{j_2}(N) \oplus \cdots \oplus PD_{j_k}(N)$, and each $PD_{j_i}(N)$, where $i = 1 \cdots k$, contains a Hamiltonian path, because each $PD_{j_i}(N)$ contains a j_i-dimensional, N^{j_i}-node grid. □

THEOREM 13. *$PT_r(N-1)$ is a subgraph of $PD_r(N)$.*

PROOF. Again using a level-order labeling of the $(N-1)$-node complete binary tree, it is well known that whenever (u, v) is an edge of the binary tree, it is also an edge of $D(N)$. The claim then follows from Theorem 3. □

COROLLARY 6. *r-dimensional mesh of $(N-1)$-node trees is a subgraph of $PD_r(N)$.*

The next two results show that $PD_r(N)$ is more powerful than the N^r-node de Bruijn graph.

THEOREM 14. *$D(N^r)$ can be embedded in $PD_r(N)$ with dilation cost r. The congestion cost is 2 when $r = 2$, or 4 when $r > 2$.*

PROOF. The case for $r = 2$ was shown in [17]. For higher dimensions, since $PD_r(N) = D(N) \oplus PD_{r-1}(N)$, a vertex of $PD_r(N)$ can be written as $u = u_{rn-1}u_{rn-2} \cdots u_{(r-1)n} \mid S'$, where S' is a vertex in $PD_{r-1}(N)$. For the discussion below, we are only interested in the leftmost bit in S', so we can write it as $S' = sS$. That is:

$$u = u_{rn-1}u_{rn-2} \cdots u_{(r-1)n} \mid sS.$$

In the N^r-node de Bruijn graph, the outgoing edges are to

$$v = u_{rn-2} \cdots u_{(r-1)n}s \mid Su_{rn-1}$$

and

$$w = u_{rn-2} \cdots u_{(r-1)n}s \mid S\overline{u}_{rn-1}.$$

For the r-dimensional product network, u has two neighbors at the highest dimension x and y, where

$$x = u_{rn-2} \cdots u_{(r-1)n}u_{rn-1} \mid sS$$

and

$$y = u_{rn-2} \cdots u_{(r-1)n}\overline{u}_{rn-1} \mid sS.$$

Let x_ℓ denote the leftmost $LogN$ bit substring of x (i.e., the part to the left of "|"). Then, observe that

$$v_\ell = w_\ell = \begin{cases} x_\ell & \text{if } u_{rn-1} = s, \\ y_\ell & \text{otherwise} \end{cases}$$

This means, by following one of the outgoing edges from u at the highest dimension, we correct the leftmost $LogN$ bits of the address towards v. Since the next set of $LogN$ bits can be corrected by the same method as above, $r - 1$ additional

steps are needed to reach v. This completes the proof that dilation of the mapping is r.

To study the congestion, first note that the first edge of the path from u to v and the first edge of the path from u to w are the same (depending on the value of s the correction of the leftmost $LogN$ bits of u takes both paths to either x or y.) Furthermore, the paths from u to v and form u to w share all the edges except for the last one, where the rightmost $LogN$ bits are corrected.

Similarly, there exist edges in $D(N^r)$ from the node

$$u' = \overline{u}_{rn-1}u_{rn-2}\cdots u_{(r-1)n} \mid sS$$

to v and w. The paths in $PD_r(N)$ from u' to v and from u' to w have a common first edge, that depending on s takes the paths to either x or y. From there they share all the remaining edges except for the last one. The paths from u to v and from u' to v share all the edges except for the first one, and same is true with the paths from u to w and from u' to w.

Therefore, the first and last edges of the paths are traversed by two of the four paths and the internal edges of the paths are traversed by the four paths identified above. Since the edges traversed by these four paths are not traversed by any other path, we can conclude that the congestion of the embedding is at most four.

When $r = 2$ the paths have length 2 and there are no internal edges, the congestion in this case is only 2. □

Earlier, it was shown in [11] that $D(2^k)$ can emulate $D(2^{k+j})$ with unit dilation cost (actually the authors of [11] called it the "4-pin shuffle graph"). In the resultant emulation, each vertex of $D(2^k)$ is assigned exactly 2^j nodes. The proof is based on the observation that, by erasing the rightmost j bits from vertex labels of $D(2^{k+j})$, we obtain a graph isomorphic to $D(2^k)$. By the same observation the following results can be stated.

COROLLARY 7. *$PD_r(N)$ can emulate $D(N^{r+j})$ with dilation cost r and congestion cost* 4 (2 *if* $r = 2$), *such that each node of $PD_r(N)$ is assigned N^j nodes.*

And moreover,

COROLLARY 8. *$PD_r(N)$ can emulate $PD_r(kN)$, where k is a power of* 2, *with unit dilation cost and unit congestion cost such that each node of $PD_r(N)$ is assigned k^r nodes.*

Therefore, for fixed r, a small size $PD_r(N)$ architecture can easily emulate larger size machines with proportional slowdown in the running time. These last two results are interesting because a small hypercube cannot emulate a larger hypercube with constant congestion. In the case of hypercube the congestion increases by the same amount as the load [20]. For de Bruijn graphs and their products, the emulation of larger graphs of their kinds require no increase in the congestion.

Finally, the next result shows that products of de Bruijn graphs are more powerful than the pure de Bruijn graphs. (This is an extension of a similar result in [17] given for two dimensions.)

THEOREM 15. *Any embedding of $PD_r(N)$ in $D(N^r)$ requires dilation $\Omega(Log(rLogN))$.*

PROOF. From Proposition 1 it suffices to show that $PD_r(N)$ contains a subgraph which cannot be embedded in $D(N^r)$ with dilation cost less than $Log(rLogN)$. From Theorem 12, we know that r-dimensional N^r-node array is a subgraph of $PD_r(N)$. It is shown in [2] that any embedding of M node k-dimensional array, for $k \geq 2$, requires dilation cost $\Omega(LogLogM)$. Since $M = N^r$, the claim follows. □

PROOF OF THEOREM 10. We know from Lemma 1 that $PS_r(N)$ and $PD_r(N)$ are computationally equivalent. If $S(N^r)$ could emulate $PS_r(N)$ with dilation less than $\Omega(Log(rLogN))$, it would imply that $S(N^r)$ could also emulate $PD_r(N)$ with dilation less than this amount, implying that shuffle-exchange network is more powerful than de Bruijn network. This contradicts with Lemma 1. □

VII. DISCUSSIONS AND CONCLUSIONS

Product networks inherently bridge the gap between many useful topologies due to their dimension-oriented definitions. It is interesting too that we are able to cite large classes of computations as being in the domain of product networks, built from a graph G, even without looking at the topology of G. For instance, the r-dimensional product of any connected graph G can emulate the r-dimensional torus with dilation cost ≤ 3 and congestion cost ≤ 2. Products of all networks which can emulate the complete binary tree with a given level of efficiency can also emulate the mesh of trees with the same level of efficiency. Additional advantages could be offered if G has other features which can be exploited at higher dimensions.

Three case studies were presented in this paper, and some of the special advantages offered by each were analyzed in detail. Fig. 10 compares different product networks with each other as well as with their nonproduct versions.

Here the columns labeled as "product" denote the $PG_r(N)$ graphs built from $G(N)$, while the columns labeled "pure" denote the $G(N^r)$ graph. The binary tree appears to benefit the most from the product definition. Its bisection width increases from 1 to $\Omega(N^{r-1})$, and the number of parallel paths increases from 1 to r. Its product version can efficiently emulate grids and mesh of trees.

In all these cases the diameter of the r-dimensional product graph is comparable with (or same as) the diameter of the corresponding size pure graph. This is because the diameters of the factor graphs studied here are logarithmic. From Theorem 1, the reader can easily check that, if $G(N^r)$ has more than logarithmic diameter, the diameter of $PG_r(N)$ must be less than that of $G(N^r)$. Conversely, if the diameter of $G(N^r)$ is less than logarithmic, the diameter of $PG_r(N)$ will be larger. There are similar relationships for the bisection width. If the bisection width of $G(N^r)$ is $\Theta(N^r)$, then this bisection width is preserved (within a constant factor) in $PG_r(N)$. Larger bisection widths are reduced, while smaller bisection widths are increased. Consideration of these factors can help predict the expected performance improvement from product definition of a given graph. On the other hand, it was shown in [6] that by crossing certain edges of the hy-

	Complete Binary Tree		Shuffle-Exchange		de Bruijn	
	product	pure	product	pure	product	pure
No. of edges	$r(N-1)N^{r-1}$	N^r-1	$(3/2)rN^r$	$(3/2)N^r$	$2rN^r$	$2N^r$
Max. vertex degree	$3r$	3	$3r$	3	$4r$	4
Min. No. of parallel paths	r	1	r	1	$2r$	4
Diameter	$2r(Log(N+1)-1)$	$2r\lceil LogN\rceil-1$	$2rLogN-r$	$2rLogN-1$	$rLogN$	$rLogN$
Bisection width	$\Omega(N^{r-1})$	1	$\Theta(\frac{N^r}{LogN})$	$\Theta(\frac{N^r}{rLogN})$	$\Theta(\frac{N^r}{LogN})$	$\Theta(\frac{N^r}{rLogN})$
Dilation/congestion of grid embedding	3/2	n/a	1/1	n/a	1/1	n/a
Dilation/congestion of MOT embedding	1/1	n/a	2/2	n/a	1/1	n/a
Notes	N^r-node binary tree is not a complete binary tree. n/a means either not known or known to be not less than logarithmic dilation. See [16]					

Fig. 10. The comparison of various properties of the three product graphs considered.

percube, the diameter can be reduced by half. A similar crossing method for product networks may be applicable also, and deserves further research.

While routing for product networks is briefly addressed (see Theorem 2), other forms of communication are not considered in this paper. A detailed investigation of various forms of data communication in product networks is a rich area awaiting investigation.

VLSI layout of product networks is briefly addressed in [17], where it is shown that two dimensional products of de Bruijn networks require a VLSI area which is more than the corresponding size de Bruijn network by a modest factor. However, there is no general result which predicts the VLSI area of $PG_r(N)$, given that VLSI area is known for $G(N)$. We addressed the VLSI layouts for the class of homogeneous product networks in a recent paper [10], but more research is needed to address several cases of heterogeneous product networks.

Finally, product networks do not have to be built with a complete $G(N)$ for each dimension. If $G(N)$ is a partitionable network, or it admits different values of N within its class definition, then it may be possible to build product networks with different sizes at different dimensions. All the investigations of data communication, VLSI area, and other relevant factors could be addressed for these networks also.

REFERENCES

[1] M. Baumslag and F. Annextein, "A unified framework for off-line permutation routing in product petworks," *Math. Systems Theory*, vol. 24, no. 4, pp. 233-251, 1991.

[2] S. Bhatt, F. Chung, J.-W. Hong, T. Leighton, and A. Rosenberg, "Optimal simulations by butterfly networks," *Proc. 20th Ann. ACM Symp. Theory of Computing*, pp. 192-204, May 1988.

[3] L.N. Bhuyan and D.P. Agrawal, "Generalized hypercube and hyperbus structures for a computer network," *IEEE Trans. Computers*, vol. 33, no. 4, pp. 323-333, Apr. 1984.

[4] M.Y. Chan, "Embedding of grids into optimal hypercubes," *SIAM J. Computing*, vol. 20, pp. 834–864, Oct. 1991.

[5] K. Efe, "Embedding mesh of trees in the hypercube," *J. Parallel and Distributed Computing*, vol. 11, no. 3, pp. 222-230, Mar. 1991.

[6] K. Efe, "The crossed cube architecture for parallel computation," *IEEE Trans. Parallel and Distributed Systems*, vol. 3, no. 5, pp. 513-524, Sept. 1992.

[7] K. Efe, "Embedding large mesh of trees and related networks on smaller hypercubes with load balancing," *Proc. Int'l Conf. Parallel Processing*, vol. 3, pp. 311–315, 1993.

[8] T. El-Ghazawi and A. Youssef, "A unified approach to fault tolerant routing," *Proc. 12th Int'l Conf. Distributed Computing Systems*, pp. 210-217, Yokohama, Japan, June 1992.

[9] R. Feldmann and P. Mysliwietz, "The shuffle exchange network has a Hamiltonian path," *Proc. Mathematical Foundations of Computer Science*, pp. 246-254, 1992.

[10] A. Fernandez and K. Efe, "Efficient VLSI layouts for homogeneous product networks," Technical Report 94-8-4, Center for Advanced Computer Studies, Univ. of Southwestern Louisiana (submitted for publication).

[11] J.P. Fishburn and R.A. Finkel, "Quotient networks," *IEEE Trans. Computers*, vol. 31 no. 4, pp. 288-295, Apr. 1982.

[12] E. Ganesan and D.K. Pradhan, "The hyper-de Bruijn multiprocessor networks: Scalable versatile architecture," *IEEE Trans. Parallel and Distributed Systems*, vol. 4, no. 9, pp. 962-978, Sept. 1993.

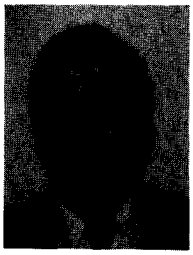
[13] I. Havel and P. Kiebl, "Embedding the polytomic tree into the n-cube," *Casopis pro Pestovan i Matematiky*, vol. 98, pp. 307-314, 1973.

[14] M. Imase, T. Soneko, and K. Okada, "Connectivity of regular directed graphs with small diameters," *IEEE Trans. Computers*, vol. 34, no. 3, pp. 267-273, Mar. 1985.

[15] R. Koch, T. Leighton, B. Maggs, S. Rao, and A. L. Rosenberg, "Work-preserving emulations for fixed-connection networks," *Proc. 21st Ann. ACM Symp. Theory of Computing*, pp. 227–240, Seattle, May 1989.

[16] F.T. Leighton, *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, and Hypercubes*. San Mateo, Calif.: Morgan Kaufmann, 1992

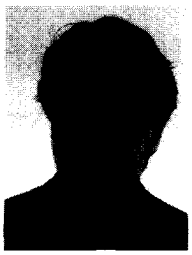
[17] A.L. Rosenberg, "Product-shuffle networks: Toward reconciling shuffles and butterflies," *Discrete Applied Mathematics*, vol. 37/38, pp. 465–488, July 1992.
[18] M. Yoeli, "Binary ring sequences," *American Math. Monthly*, vol. 69, pp. 852-855, 1962.
[19] C.D. Thompson, "A complexity theory for VLSI," PhD thesis, Carnegie-Mellon Univ., Aug. 1980.
[20] A.K. Gupta, A.J. Boals, N.A. Sherwani, and S.E. Hambrusch, "A lower bound on embedding large hypercubes into small hypercubes," *Congressus Numerantium*, vol. 78, pp. 141-151, 1990.



Kemal Efe received the BSc degree in electronic engineering from Istanbul Technical University, the MS degree in computer science from UCLA, and the PhD degree in computer science from the University of Leeds.

He is currently an associate professor of computer science in the Center for Advanced Computer Studies, University of Southwestern Louisiana, Lafayette. Previously, he was on the faculty of the Computer Science Department, University of Missouri-Columbia.

Dr. Efe's research interests are in parallel and distributed computing, in which he has authored more than 50 refereed papers. His expertise includes parallel architectures and algorithms, interconnection networks, distributed operating systems, performance evaluation, and algorithms for loosely coupled workstation networks. Dr. Efe served on the technical committees of several conferences and gave invited talks in the U.S. and Europe. He is a member of the ACM and the IEEE.



Antonio Fernández received the degree of Diplomado en Informática in March 1988 and the degree of Licenciado en Informática in July 1991 from the Universidad Politécnica de Madrid. He received the MS degree in computer science in the fall of 1992 and the PhD degree in computer science in the fall of 1994 from the University of Southwestern Louisiana, supported by a Fulbright Scholarship.

He is an associate professor in the Departamento de Arquitectura y Tecnología de Computadores at the Universidad Politécnica de Madrid, where he has served on the faculty since 1988. He is currently on leave as a post-doctoral researcher at the Laboratory for Computer Science at MIT.

Dr. Fernández's research interests include parallel architectures and algorithms, interconnection networks, distributed systems, and data communication.