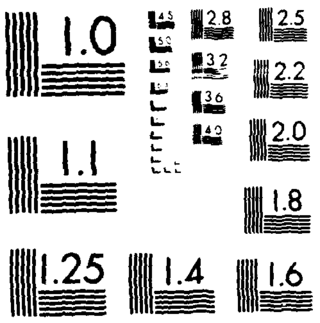
AD-A195 168 OPTIMAL SIMULATIONS BY BUTTERFLY NETWORKS: EXTENDED ABSTRACT(U) MASSACHUSETTS INST OF TECH CAMBRIDGE S N BHATT ET AL. NOV 87 N00014-80-C-0622 1/1

UNCLASSIFIED F/G 25/5 NL

END
DATE
FILMED
8 88

[He] L.S. Heath and A.L. Rosenberg (1987): An optimal mapping of the FFT algorithm onto the Hypercube architecture. Typescript, Univ. of Massachusetts; submitted for publication.


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0

MASSACHUSETTS INSTITUTE OF TECHNOLOGY VLSI PUBLICATIONS

VLSI Memo No. 87-427
November 1987

OPTIMAL SIMULATIONS BY BUTTERFLY NETWORKS: EXTENDED ABSTRACT

Sandeep N. Bhatt, Fan R. K. Chung, Jia-Wei Hong, F. Thomas Leighton and Arnold Rosenberg

Abstract

We investigate the power of the Butterfly network (which is the FFT network with inputs and outputs identified) relative to other proposed multicomputer interconnection networks, by considering how efficiently the Butterfly can simulate the other networks: Formally we ask, How efficiently can one embed the graph underlying the other network in the graph underlying the Butterfly? We measure the efficiency of an embedding of a graph G in a graph H in terms of: the *dilation*, or, the maximum amount that any edge of G is "stretched" by the embedding; the *expansion*, or, the ratio of the number of vertices of H to the number of vertices of G. We present three simulations that are optimal, to within constant factors: (1) Any complete binary tree can be embedded in a Butterfly graph, with simultaneous dilation $O(1)$ and expansion $O(1)$. (2) The n-vertex X-tree can be embedded in a Butterfly graph with simultaneous dilation $O(\log \log n)$ and expansion $O(1)$; no embedding has better dilation, independent of expansion. (3) Any embedding of the $n \times n$ mesh in the Butterfly graph must have dilation $(\log n)$, independent of expansion; any embedding of the mesh in the Butterfly graph achieves this dilation. Thus, we have simulations of complete-binary-tree machines, X-tree machines, and mesh computers on Butterfly machines, that are optimal in resource utilization (expansion) and delay (dilation), to within constant factors.

Microsystems Research Center Room 39-321
Massachusetts Institute of Technology
Cambridge Massachusetts 02139
Telephone (617) 253-8138

Acknowledgements

This work was supported in part by the Air Force under contract number AFOSR-86-0076, the Defense Advanced Research Projects Agency under contract number N00014-80-C-0622, the MIT Army Center for Intelligent Control, and an NSF Presidential Young Investigator Award with matching funds from AT&T and IBM.

Author Information

Bhatt: Department of Computer Science, Yale University, New Haven, CT 06520; Chung: Mathematics, Information Sciences and Operations Research Division, Bell Communications Research, Morristown, NJ 07960; Hong: Beijing Computer Institute, Beijing 10044, CHINA and Courant Institute of Mathematics, New York University, New York, NY 10012; Rosenberg: Department of Computer and Information Science, University of Massachusetts, Amherst, MA 01003; Leighton: Department of Mathematics, MIT, Room NE43-419, Cambridge, MA 02139, (617)253-6043.

OPTIMAL SIMULATIONS BY BUTTERFLY NETWORKS: Extended Abstract

Sandeep N. Bhatt[*] *Fan R. K. Chung*[†] *Jia-Wei Hong*[‡]
F. Thomson Leighton[§] *Arnold L. Rosenberg*[¶]

October 7, 1987

Abstract

We investigate the power of the Butterfly network (which is the FFT network with inputs and outputs identified) relative to other proposed multicomputer interconnection networks, by considering how efficiently the Butterfly can simulate the other networks: Formally, we ask, How efficiently can one embed the graph underlying the other network in the graph underlying the Butterfly? We measure the efficiency of an embedding of a graph G in a graph H in terms of: the *dilation*, or, the maximum amount that any edge of G is "stretched" by the embedding; the *expansion*, or, the ratio of the number of vertices of H to the number of vertices of G. We present three simulations that are optimal, to within constant factors: (1) Any complete binary tree can be embedded in a Butterfly graph, with simultaneous dilation $O(1)$ and expansion $O(1)$. (2) The n-vertex X-tree can be embedded in a Butterfly graph with simultaneous dilation $O(\log\log n)$ and expansion $O(1)$; no embedding has better dilation, independent of expansion. (3) Any embedding of the $n\times n$ mesh in the Butterfly graph must have dilation $\Omega(\log n)$, independent of expansion; any embedding of the mesh in the Butterfly graph achieves this dilation. Thus, we have simulations of complete-binary-tree machines, X-tree machines, and mesh computers on Butterfly machines, that are optimal in resource utilization (expansion) and delay (dilation), to within constant factors.

[*]Department of Computer Science, Yale University, New Haven, CT 06520
[†]Mathematics, Information Sciences and Operations Research Division, Bell Communications Research, Morristown, NJ 07960
[‡]Beijing Computer Institute, Beijing 10044, CHINA and Courant Institute of Mathematics, NYU, New York, NY 10012
[§]Department of Mathematics, MIT, Cambridge, MA 02139
[¶]Department of Computer and Information Science, University of Massachusetts, Amherst, MA 01003

1. INTRODUCTION

This paper reports on a continuing program of the authors, dedicated to determining the relative computational capabilities of the various interconnection networks that have been proposed for use as multicomputer interconnection networks [BC, BI, He, Le]. We focus here on one member of the family of ***butterfly*** machines, that have become one of the benchmark architectures for multicomputers.

1.1. The Formal Setting

The technical vehicle for our investigations is the following notion of graph embedding. Let G and H be simple undirected graphs. An ***embedding*** of G in H is a one-to-one association of the vertices of G with vertices of H. The ***dilation*** of the embedding is the maximum distance (in H) between vertices of H that are the images of adjacent vertices of G; it thus measures how much the edges of G are "stretched" by the embedding. The ***expansion*** of the embedding is the ratio $|H|/|G|$ of the number of vertices in H to the number of vertices in G. We use the dilation- and expansion-costs of the best embedding of G in H as our measures of how well H can *simulate* G as an interconnection network: One views the graph H as abstracting the processor-intercommunication structure of a physical architecture; one views the graph G as abstracting either the task-interdependency structure of an algorithm one wants to implement on H or the processor-intercommunication structure of an architecture one wants to simulate on H.

The host graphs in our study, which play the role of our H's, are ***Butterfly networks*** (that is, FFT networks whose input and output vertices have been identified). Formally,

> Let m be a positive integer. The ***m-level Butterfly graph*** $B(m)$ has vertex-set
>
> $$V_m = \{0, 1, \cdots, m-1\} \times \{0,1\}^m,$$
>
> where $\{0,1\}^m$ denotes the set of length-m binary strings. The subset $V_{m,\ell} = \{\ell\} \times \{0,1\}^m$ of V_m $(0 \le \ell < m)$ is the ℓ^{th} ***level*** of $B(m)$. The string $x \in \{0,1\}^m$ of vertex $\langle \ell, x \rangle$ is the ***position-within-level string*** (***PWL string***, for short) of the vertex. The edges of $B(m)$ form ***butterflies*** (or, copies of $K_{2,2}$) between consecutive levels of vertices, with wraparound in the sense that level 0 is identified with level m. Each butterfly connects vertices
>
> $$\langle \ell,\ \beta_0\beta_1 \cdots \beta_{\ell-1}0\beta_{\ell+1} \cdots \beta_{m-1} \rangle$$
>
> and
>
> $$\langle \ell,\ \beta_0\beta_1 \cdots \beta_{\ell-1}1\beta_{\ell+1} \cdots \beta_{m-1} \rangle$$
>
> on level ℓ of $B(m)$ $(0 \le \ell < m$; each $\beta_i \in \{0,1\})$ with vertices
>
> $$\langle \ell + 1 (\text{mod } m),\ \beta_0\beta_1 \cdots \beta_{\ell-1}0\beta_{\ell+1} \cdots \beta_{m-1} \rangle$$

and

$$\langle \ell + 1 (\text{mod } m),\ \beta_0\beta_1 \cdots \beta_{\ell-1} 1 \beta_{\ell+1} \cdots \beta_{m-1} \rangle$$

on level $\ell + 1(\text{mod } m)$ of $B(m)$. One can represent $B(m)$ level by level, in such a way that at each level the PWL strings are the reversals of the binary representations of the integers $0, 1, \ldots, m-1$, in that order.

The guest graphs in our study, which play the role of our G's, are the height-h complete binary tree $T(h)$, the height-h X-tree $X(h)$ (which is obtained from $T(h)$ by adding *cross edges* connecting the vertices at each level of $T(h)$ in a path, with the vertices in left-to-right order), and the $s \times s$ mesh $M(s)$. All of these networks have been seriously proposed as interconnection networks for multicomputers [DP, Ga, HZ], hence are important candidates for our study. Another approach to comparing these networks, via implementation and analysis of specific algorithms, appears in [Ag].

1.2. The Main Results

Theorem 1 *The complete binary tree $T(h)$ can be embedded in a Butterfly graph, with simultaneous dilation $O(1)$ and expansion $O(1)$.*

Theorem 1 is within a constant factor of optimal in both dilation and expansion.

Theorem 2 *The height-h X-tree $X(h)$ can be embedded in a Butterfly graph with simultaneous dilation $O(\log h) = O(\log\log|X(h)|)$ and expansion $O(1)$. Any embedding of $X(h)$ in a Butterfly graph must have dilation $\Omega(\log h) = \Omega(\log\log|X(h)|)$.*

Theorem 3 *Any embedding of the side-s mesh $M(s)$ in a Butterfly graph must have dilation $\Omega(\log s) = \Omega(\log|M(s)|)$.*

Theorem 3 betokens a mismatch in the structures of meshes and Butterfly graphs, since *any* expansion-$O(1)$ embedding of any graph G in $B(m)$ has dilation $O(\log|G|)$.[1]

In the next three sections we prove the main results. We sketch the upper bound from Theorem 1 and the lower bound from Theorem 2 in some detail, to indicate the machinery we bring to bear on the problem; other proofs are merely outlined.

The sketched proofs are important because they can be adapted to other families of graphs: For instance, our embedding of complete binary trees can be used to embed arbitrary bounded-degree outerplanar graphs in Butterflies with simultaneous dilation $O(\log\log n)$ and expansion $O(1)$, in much the way that we embed $X(h)$; and, our lower bound on embeddings of X-trees can be adapted to yield analogous bounds for any family of "mesh-like" graphs, as defined in Section 3.2, in much the way that we obtain our lower bound for meshes.

[1]This follows from the facts that $B(m)$ has $m2^m$ vertices and diameter $O(m)$.

2. COMPLETE BINARY TREES

2.1. Optimally Embedding Trees in Butterfly Graphs

To simplify exposition, we represent sets of binary strings by strings over the alphabet $\{0, 1, *\}$, using $*$ as a wild-card character. The length-k string

$$\beta = \beta_0\beta_1 \cdots \beta_{k-1},$$

each $\beta_i \in \{0, 1, *\}$, represents the set $\sigma(\beta)$ of all length-k binary strings that have a 0 in each position i where $\beta_i = 0$, a 1 in each position i where $\beta_i = 1$, and either a 0 or a 1 in each position i where $\beta_i = *$. Call β the *code* for the set $\sigma(\beta)$.

A. The Embedding Strategy

Let m be a power of 2; clerical changes will remove this assumption. We wish to embed the tree $T(m + \log m)$ with dilation $O(1)$, in the smallest Butterfly that is big enough to hold the tree, namely, $B(m)$. We fall somewhat short of this goal, but not by much: We find an embedding with dilation $O(1)$, but we have to use a somewhat larger host Butterfly graph (specifically, $B(m+3)$) in order to resolve collisions in our embedding procedure. Our embedding proceeds in four stages. Stage 1 embeds the top $\log m$ levels of $T(m + \log m)$ with unit dilation in $B(m)$, thereby specifying implicitly the images in $B(m)$ of the roots of the m subtrees of $T(m + \log m)$ rooted at level $\log m$. Stage 2 expands these m subtrees a further $m/2$ levels, but now in $B(m + 1)$, with dilation 2, thereby specifying implicitly the images in $B(m)$ of the roots of the $m \cdot 2^{m/2}$ subtrees of $T(m + \log m)$ rooted at level $m/2 + \log m$ of the tree. In Stage 3, we embed the final $m/2$ levels of $T(m + \log m)$ in $B(m + 1)$, with dilation 2. The vertex-mappings in each stage are embeddings (i.e., are one-to-one); there is, however, "overlap" (i.e., distinct vertices of $T(m + \log m)$ getting mapped to the same vertex of $B(m + 1)$) among the mappings of the three stages. In Stage 4, we eliminate this overlap by expanding the host Butterfly by two more levels, thereby giving us four connected isomorphic copies of $B(m + 1)$. At the cost of increasing dilation to 4, we modify our mapping so that each of Stages 1, 2, 3 is performed in a distinct copy of $B(m + 1)$, thereby eliminating all overlap.

B. Stage 1: Levels 1 - $\log m$ of $T(m + \log m)$

We place the root of $T(m + \log m)$ at position

$$\langle m - \log m,\ 0^m \rangle$$

of $B(m)$. We then proceed to higher-numbered levels, embedding the top $\log m$ levels of $T(m + \log m)$ as a subgraph of $B(m)$, ending up with the leaves of these levels in positions

$$\langle 0,\ 0^{m-\log m} *^{\log m} \rangle$$

of $B(m)$ (because of wraparound). We call the rightmost $\log m$ bits of each of the resulting PWL strings the *signature* of the Butterfly position and of the subtree rooted at that position. It is convenient to interpret a signature as an integer in the range $\{0, 1, \cdots, m-1\}$, as well as a bit string.

The embedding in Stage 1 is trivially one-to-one, with unit dilation.

C. Stage 2: Levels $\log m$ - $(m/2 + \log m)$ of $T(m + \log m)$

Call the subtree of $T(m+\log m)$ with signature k, the k^{th} ***subtree***. Our goal is to embed the k^{th} subtree in $B(m+1)$ (with dilation 2), so that its $2^{m/2}$ leaves form the set of positions

$$\langle m-1, \ *0*0\cdots*0*1*0\cdots*0*0*\rangle,$$

where the 1 appears in the k^{th} even position from the right (using 0-based counting); call this the ***signatory*** 1 of the tree position.[2] We embed these $(m/2)$-level trees by alternating binary and unary branchings in $B(m+1)$, starting at the "roots" placed at level-0 vertices of $B(m+1)$ during Stage 1; we place a tree vertex after each unary branching. Binary branchings generate the $*$'s in the code for the set of PWL strings, while unary branchings generate the 0's and 1's in the code.

This stage of our embedding clearly has dilation 2. One notes that this stage is one-to-one (though it may produce conflicts with the embedding from Stage 1), since the signatory 1, whose placement identifies the tree, is set "on" before the signature bits are reached and altered by the sequence of branchings. This is ensured by the fact that we place the signatory 1 by counting from the right: the signature bits occupy the rightmost $\log m$ bits of the PWL string; by the time the branchings have reached the i^{th} bit from the right, only the rightmost $\log i$ bits of the signature are needed to specify the next position where branching occurs. Hence, at the point when we place the signatory 1 in the i^{th} position, the odd-numbered positions to the left of the 1 are all 0, and the positions to the right of the 1 form the binary representation of i, possibly with leading 0's.

D. Stage 3: Levels $(m/2 + \log m)$ - $(m + \log m)$ of $T(m + \log m)$

Our goal in Stage 3 is to use the leaves of the trees generated in Stage 2 as the roots of the $m \cdot 2^{m/2}$ subtrees comprising the bottom $m/2$ levels of $T(m + \log m)$. Each root has a signatory 1, identifying the subtree it came from in Stage 2, and a ***serial number*** obtained from the even-numbered bits of its PWL string. The signatory 1 helped to keep the many trees generated in Stage 2 disjoint; the signatory 1 plus the serial number play the same role here. The main difficulty here is to achieve the embedding while the roots of all the trees reside at the same level of $B(m+1)$ (which is how Stage 2 has placed them). To accomplish this, we have the trees grow ***upward***, in the direction of lower level-numbers, for varying amounts of time, before starting to grow ***downward***, in the direction of higher level-numbers. Whether a tree grows upward or downward, it grows via alternating unary and binary branchings; when it "turns" from growing upward to growing downward, the embedding has to stretch edges by an extra factor of 2, as it moves from one subgraph of $B(m+1)$ to a neighboring isomorphic subgraph.

[2]For instance, when $m = 8$, the second subtree has leaves in positions $\langle 7, \ *0*1*0*0*\rangle$ of $B(9)$.

To simplify our description, we ignore the serial numbers of the trees and present just the other $m/2$ bits of each PWL string. We use alternating unary and binary branchings when placing tree vertices to guarantee that serial numbers do not change throughout the embedding procedure. This ensures the mutual disjointness of all trees that share the same signatory 1, albeit at the cost of increasing dilation by a factor of 2. (The position of the signatory 1 ensures the mutual disjointness of trees with different signatory 1's.)

At each level ℓ of $B(m+1)$, $m+2-k \leq \ell \leq m+1$, we reserve the following positions for vertices from the kth subtree, $0 \leq k < m/2$.

$$\langle \ell,\ 0^{m/2-k}1*^{k-1}\rangle.$$

We place the top $k-1$ levels of the kth subtree in the reserved positions via the following scheme. At level 0 of $B(m+1)$, the root of the subtree is placed in position

$$\langle 0,\ 0^{m/2-k}10^{k-1}\rangle;$$

and the 2^{k-2} vertices from level $k-2$ of the subtree (for $k \geq 2$) are placed at positions

$$\langle 0,\ 0^{m/2-k}11*^{k-2}\rangle.$$

At each level $m-2j$ of $B(m+1)$, $1 \leq j \leq \lfloor (k-3)/2 \rfloor$, the 2^j level-j vertices of the subtree are placed in positions

$$\langle m-2j,\ 0^{m/2-k}10^{k-1-j}*^{j}\rangle;$$

and the 2^{k-2-j} vertices from level $k-2-j$ of the subtree are placed at positions

$$\langle \ell,\ 0^{m/2-k}11*^{k-2-j}1^{j}\rangle.$$

When k is even, one more level of $B(m+1)$ is needed, for the middle level of the subtree. This portion of the embedding engenders dilation 4, namely, dilation 2 for the alternating unary and binary branchings and dilation 2 for the top level, where the subtree stops growing upward and starts growing downward.

Proceeding down $B(m+1)$, from level 0 to higher-numbered levels, we reserve the following positions for vertices from the kth subtree at level ℓ, $1 \leq \ell \leq m/2-k$.

$$\langle \ell,\ *^{\ell}0^{m/2-k-\ell}10^{k-1}\rangle.$$

The tree gets expanded into these positions, basically as a subgraph except for the skipping of alternate levels in order to preserve the serial number. This portion of the embedding engenders dilation 2.

The injectiveness and the dilation of the described mapping should be clear.

E. Resolving Collisions

We resolve the possible collisions among our three injective subembeddings as follows. Instead of performing the embeddings in $B(m+1)$, we perform them in $B(m+3)$, placing each subembedding in a distinct copy of $B(m+1)$. We make the transition between copies of $B(m+1)$ as follows. As the Stage-1 embedding of the top of $T(m+\log m)$ reaches level $m-1$ of $B(m+1)$, we use a sequence of unary branchings in $B(m+3)$ to reach level 0 of the next copy of $B(m+1)$. We perform the Stage-2 subembedding within this copy, which takes us to level $m-1$ of the copy, where a sequence of unary branchings in $B(m+3)$ takes us to level 0 of the third copy of $B(m+1)$. We perform the Stage-3 subembedding in this third copy. This Stage of the embedding process engenders dilation 4, as one switches from level $m-1$ of the second copy of $B(m+1)$ to level 0 of the third copy.

The embedding, hence the proof, is now complete. □

2.2. The Issue of Optimality

Theorem 1 leaves open the possibility of constant-factor improvements, but it is impossible to optimize both dilation and expansion simultaneously.

Proposition 1 *No embedding of* $T(m+\lfloor\log m\rfloor)$ *in* $B(m)$ *has unit dilation.*

Proof. Both $T(h)$ and $B(m)$ are bipartite. In $B(m)$, the numbers of red and blue vertices are within m of equal; in $T(h)$, one of the sets is roughly twice as large as the other. □

3. X-TREES

3.1. Optimally Embedding X-Trees in Butterfly Graphs

Our embedding of the X-tree in the Butterfly graph is indirect: First we find a unit-expansion, dilation-$O(\log\log n)$ embedding of $X(h)$ in $T(h)$. Then we compose this embedding with the expansion-$O(1)$, dilation-$O(1)$ embedding of $T(h)$ in $B(m)$ from Theorem 1, to obtain the upper bound of Theorem 2. We discuss here only the former embedding, which uses a device from [BC, BL]. Details appear in the full paper.

It is not hard to verify that for each positive integer k, the n-vertex X-tree has a *k-color bisector* [BL] of size $k\cdot\log^2 n$.[3] Using 3-color bisectors in the same manner as in [BC], one proves the following.

Lemma 1 *There is a* many-to-one *mapping of* $X(h)$ *into* $T(h)$ *for which*
(a) *exactly*

$$N(\ell)=12\log^2\left(\frac{2^{h+1}-1}{2^\ell}\right)+216\log\left(\frac{2^{h+1}-1}{2^\ell}\right)+2268$$

[3]That is, for every way of labelling $X(h)$'s vertices with one of k "colors": By removing $\leq k\cdot\log^2 n$ vertices, one can partition $X(h)$ into two subgraphs, each having half of the vertices of each color.

vertices of $X(h)$ are mapped to each level-ℓ vertex of $T(h)$, and
(b) *the images of vertices that are adjacent in $X(h)$ are at most distance 3 apart in $T(h)$.*

We refine the "dilation"-3 mapping of Lemma 1 to a dilation-$O(\log\log n)$ embedding of $X(h)$ in a new copy of $T(h)$, inductively. First we spread the $\log n$ elements at the root of the old copy of $T(h)$ throughout the topmost $\log\log n$ levels of the new copy of $T(h)$, in any way. In rough terms, we then spread the X-tree vertices mapped onto level-($\ell > 0$) vertices of the old copy of $T(h)$ across level $\ell + \log\log n$ of the new copy of $T(h)$, retaining the order induced by their position in level ℓ of the old tree.

This procedure clearly produces an embedding of $X(h)$ in $T(h)$; the embedding has unit expansion since all vertices at each level ℓ of the old tree have the same population $N(\ell)$; the embedding has dilation $O(\log\log n)$ since our method of spreading the images from vertices in the old tree guarantees that the spread vertices always lie within a subtree of height $O(\log\log n)$ in the new copy of $T(h)$. □

3.2. The Optimality of the Embedding

We demonstrate the optimality (to within constant factors) of the previous embedding, by proving the lower bound of Theorem 2. Our proof technique extends directly to a broad class of graphs that we call "mesh-like". Assume henceforth that we are given a dilation-δ embedding μ of a graph G in the Butterfly graph $B(p)$.

Say that a graph H is *c-weakly connected*, c a positive integer, if every two vertices u, w of H are connected by a chain of vertices:

$$u = v_0, v_1, v_2, \ldots, v_k = w,$$

where consecutive vertices v_i, v_{i+1} are at distance no more than c apart, in H;

Proposition 2 *Say that there is a subgraph H of G and constants c and d such that*

- *H is c-weakly connected;*
- *the image of H under the embedding μ lies within $d\delta$ consecutive levels of $B(p)$.*

Then $\delta \geq \alpha(c)\log|H|$, where $\alpha(c)$ is a constant depending only on c.

Proof Sketch. One can show that the PWL strings of all images of vertices of H can differ in some set of at most $(2c+1)\delta$ bit positions; hence, there can be no more than $d\delta 2^{(2c+1)\delta}$ such vertices ($d\delta$ levels and $2^{(2c+1)\delta}$ vertices per level), so this number can be no smaller than $|H|$. □

A *slice* of a graph G is a set of vertices whose removal partitions G into subgraphs, each having $\leq |G|/3$ vertices. Let S_G denote the number of vertices in the smallest slice of G.

We say that the graph G is *mesh-like* if every simply connected component of G of size less than S_G has a 2-weakly connected boundary[4].

Proposition 3 *If the graph G is mesh-like, then the embedding μ must have dilation $\delta \geq (const) \log S_G$.*

Proof Sketch. Partition $B(p)$ into *bands*, each band β_i being a sequence of $c_i\delta$ consecutive levels, $2 \leq c_i < 4$, where the c_i can be chosen in any way that achieves a partition. Let $\kappa(v)$, the *color* of vertex v of G, be the index i of the band β_i in which $\mu(v)$ resides.

We perform a modified breadth-first search of G, to find a 2-weakly connected component of size $\geq S_G$, all of whose vertices have images in a single band of $B(p)$, hence the same color. By Proposition 2, the existence of such a component will yield the lower bound on δ.

The breadth-first search proceeds as follows. We select an arbitrary vertex v_0 of G. Let V_0 be the maximal connected component of G that contains v_0 and that consists entirely of vertices with color $\kappa(v_0)$. Since V_0 is (2-weakly) connected, removing its vertices from G separates G into at least two connected components; let C_0 be the largest of these. Form the *simple hull* C_0^* of C_0.[5] Let B_0 be that subset of C_0^* that is part of the boundary of V_0. Since G is mesh-like, B_0 is 2-weakly connected. One can show, therefore, that all vertices of B_0 have the same color (which is either $\kappa(v_0) + 1$ or $\kappa(v_0) - 1$). Now, let V_1 be the maximal monochromatic subgraph of G that contains both B_0 and all connected components of C_0^* that contain vertices from B_0; obviously, V_1 is 2-weakly connected. In an analogous way and using analogous reasoning, we now construct, in turn, for $i = 1, 2, \ldots$, the following (monochromatic) subgraphs of G, with the indicated properties:

- C_i: the largest connected component of G remaining when one removes V_i
- C_i^*: the simple hull of C_i
- B_i: the (2-weakly connected, monochromatic) boundary of V_i within C_i^*
- V_{i+1}: the (2-weakly connected) maximal monochromatic subgraph of G that contains both B_i and all connected components of C_i^* that contain vertices from B_i.

One continues this construction until some subgraph V_i contains at least S_G vertices. This point must occur, since one can show that each V_i is a cutset of G, and that the largest components remaining upon removal of V_i decrease in size as i increases. It follows that at

[4]The *boundary* comprises those vertices that are not in the simply connected component but are adjacent to vertices that are in the component.

[5]C_0^* is the smallest simply connected supergraph of C_0.

some point, all of the components remaining upon removal of some V_i will contain at most $|G|/3$ vertices. The then-current V_i will be a slice of G, hence will be a weakly-connected, monochromatic set of at least S_G vertices. At this point, we shall be done. □

The proof is completed by the following two lemmas.

Lemma 2 *The X-tree $X(h)$ is mesh-like.*

Lemma 3 [HR] *Any slice of $X(h)$ must contain $\Omega(h) = \Omega(\log|X(h)|)$ vertices.*

4. MESHES

The upper bound being trivial, we establish only the lower bound, which follows from Proposition 3 and the following two lemmas.

Lemma 4 *The mesh $M(s)$ is mesh-like.*

Lemma 5 (e.g., [HR]) *Any slice of $M(s)$ must contain $\Omega(s) = \Omega(\sqrt{|M(s)|})$ vertices.*

ACKNOWLEDGMENT: The research of A. L. Rosenberg was supported in part by NSF Grant DCI-87-96236.

5. REFERENCES

[**Ag**] A. Aggarwal (1984): A comparative study of X-tree, pyramid, and related machines. *25th IEEE Symp. on Foundations of Computer Science*, 89-99.

[**BC**] S.N. Bhatt, F.R.K. Chung, F.T. Leighton, A.L. Rosenberg (1986): Optimal simulations of tree machines. *27th IEEE Symp. on Foundations of Computer Science*, 274-282.

[**BI**] S.N. Bhatt and I. Ipsen (1985): Embedding trees in the hypercube. Yale Univ. Rpt. RR-443.

[**BL**] S.N. Bhatt and F.T. Leighton (1984): A framework for solving VLSI graph layout problems. *J. Comp. Syst. Sci. 28,* 300-343.

[**DP**] A.M. Despain and D.A. Patterson (1978): X-tree - a tree structured multiprocessor architecture. *5th Symp. on Computer Architecture,* 144-151.

[**Ga**] D. Gannon (1980): On pipelining a mesh-connected multiprocessor for finite element problems by nested dissection. *Intl. Conf. on Parallel Processing.*

[He] L.S. Heath and A.L. Rosenberg (1987): An optimal mapping of the FFT algorithm onto the Hypercube architecture. Typescript, Univ. of Massachusetts; submitted for publication.

[HR] J.-W. Hong and A.L. Rosenberg (1982): Graphs that are almost binary trees. *SIAM J. Comput. 11*, 227-242.

[HZ] E. Horowitz and A. Zorat (1981): The binary tree as an interconnection network: applications to multiprocessor systems and VLSI. *IEEE Trans. Comp., C-30*, 247-253.

[Le] F.T. Leighton (1984): Parallel computation using meshes of trees. *1983 Workshop on Graph-Theoretic Concepts in Computer Science*, Trauner Verlag, Linz, pp. 200-218.

8