AN OPTIMAL MAPPING OF THE FFT ALGORITHM ONTO THE HYPERCUBE ARCHITECTURE

Lenwood S. Heath
Arnold L. Rosenberg

COINS Technical Report 87-19

AN OPTIMAL MAPPING OF THE FFT ALGORITHM ONTO THE HYPERCUBE ARCHITECTURE

Lenwood S. Heath
Dept. of Mathematics
MIT
Cambridge, MA 02139

Arnold L. Rosenberg
Dept. of Computer and Information Science
University of Massachusetts
Amherst, MA 01003

March 6, 1987

Abstract

We prove that each FFT graph is a subgraph of the smallest Boolean hypercube that is big enough to hold it, and we present a linear-time algorithm that finds the embedding. Thus, we find a mapping of the FFT algorithm onto the Hypercube architecture, with unit (hence, optimal) dilation and optimal expansion.

1. INTRODUCTION

We consider a problem that arises when one implements a parallel algorithm on an array of processors. On the one hand, the algorithm has some natural subtask-interdependence structure; on the other hand, the array has a fixed processor-intercommunication network. The *mapping problem* for the desired algorithm and the available processor array is the problem of accommodating the algorithm's interdependence structure to the array's intercommunication structure. One typically studies the mapping problem by viewing both of the structures of interest as simple undirected graphs and viewing the accommodation problem as one of finding an efficient embedding of the algorithm-graph in the array-graph [2, 3, 4, 5]. The major notions of the efficiency of an embedding are enunciated in [8]; they are the *dilation* of the embedding, which measures the maximum delay engendered by the accommodation, and the *expansion* of the embedding, which is one measure of the efficiency of utilizing the processors of the array.

In this paper, we study the mapping problem for the important *Fast Fourier Transform* (*FFT*) algorithm [1, Ch. 7], which is paradigmatic for convolution-based algorithms, and the popular (Boolean) Hypercube architecture [3, 4, 6], versions of which have been built by Intel, N-cube, BBN, and Thinking Machines. Our main result is a linear-time algorithm that embeds the intercommunication graph of the FFT algorithm in the Hypercube network,

with unit (hence, optimal) dilation and optimal expansion. This result provides yet another example of the efficiency of the Hypercube as an interconnection structure, to supplement earlier work that has shown the Hypercube to be an efficient host for divide-and-conquer algorithms [3] and for grid-based algorithms [7].

2. THE FORMAL FRAMEWORK

The technical vehicle for our investigation is the following notion of graph embedding. Let G and H be simple undirected graphs, having $|G|$ vertices and $|H|$ vertices, respectively. An *embedding* of G in H is a one-to-one association of the vertices of G with the 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. The *expansion* of the embedding is the ratio $|H|/|G|$. Clearly, no embedding can have better than unit dilation, and such dilation is achievable only if G is a subgraph of H.

Our specific focus here will be on embeddings between two given finite families of graphs Γ and Φ. We shall be seeking the best possible embeddings – relative to dilation-cost – of each $G \in \Gamma$ in the smallest $H \in \Phi$ that will hold it, i.e., for which $|H|/|G| \geq 1$. Thus we optimize expansion-cost and then try to optimize dilation-cost. We shall be able here to achieve unit (hence, optimal) dilation, even while optimizing expansion.

The target graphs for our investigation are FFT graphs (which will play the role of our G's) and Boolean hypercubes (which will play the role of our H's).

- Let m be a positive integer. The 2^m-*input FFT graph* (so named because it reflects the data-dependency structure of the 2^m-input FFT algorithm; cf. [1, Ch. 7]), denoted $F(m)$, is defined as follows. $F(m)$ has vertex-set

$$V_m = \{0, 1, \cdots, m\} \times \{0, 1, \cdots, 2^m - 1\}$$

The subset $V_{m,\ell} = \{\ell\} \times \{0, 1, \cdots, 2^m - 1\}$ of V_m $(0 \leq \ell \leq m)$ is called the ℓ^{th} *level* of $F(m)$; vertices in $V_{m,0}$ are called *inputs*, and vertices in $V_{m,m}$ are called *outputs* (in deference to the algorithmic origin of the graph). The edges of $F(m)$ form *butterflies* (or, copies of the complete bipartite graph $K_{2,2}$) between consecutive levels of vertices. Each butterfly connects vertices

$$\langle \ell, \alpha 2^{\ell+1} + \beta \rangle \text{ and } \langle \ell, \alpha 2^{\ell+1} + \beta + 2^\ell \rangle$$

on level ℓ of $F(m)$ $(0 \leq \ell < m;\ 0 \leq \alpha < 2^{m-\ell-1};\ 0 \leq \beta < 2^\ell)$ with vertices

$$\langle \ell + 1, \alpha 2^{\ell+1} + \beta \rangle \text{ and } \langle \ell + 1, \alpha 2^{\ell+1} + \beta + 2^\ell \rangle$$

on level $\ell + 1$. It is often useful to view $F(m)$, $m \geq 2$ ($F(1) = K_{2,2}$ being given), as being constructed inductively, by taking two copies of $F(m-1)$, and 2^m new output vertices, and constructing butterflies connecting the k^{th} outputs of each copy of $F(m-1)$, on the one side, to the k^{th} and $(k + 2^{m-1})^{\text{th}}$ new outputs, on the other side. Thus, $F(m)$ has $(m+1)2^m$ vertices and $m2^{m+1}$ edges.

- Let d be a nonnegative integer. The *d-dimensional Boolean Hypercube* $C(d)$ is the graph whose vertices are all binary strings of length d and whose edges connect each string-vertex x with the d strings that differ from x in precisely one position. Thus, $C(d)$ has 2^d vertices and $d2^{d-1}$ edges.

It is not hard to find a unit-dilation embedding of $F(m)$ in $C(2m)$, by assigning two new dimensions for each level of $F(m)$; but this embedding has expansion $\Omega(N/\log N)$. Likewise, it is not hard to find a dilation-2 (expansion-optimal) embedding of $F(m)$ in $C(m + \lceil \log_2(m+1) \rceil)$, which is the smallest Hypercube that holds $F(m)$, using embedding techniques analogous to those used in [4]. The main result of this paper is a linear-time algorithm that finds a unit-dilation embedding of $F(m)$ in $C(m + \lceil \log_2(m+1) \rceil)$, thereby optimizing dilation and expansion via the same embedding. Stated formally, we prove

Theorem 1 *The FFT graph $F(m)$ is a subgraph of $C(m + \lceil \log_2(m+1) \rceil)$; moreover, there is a linear-time algorithm that finds this optimal embedding.*

3. THE EMBEDDING

We now describe the embedding, and implicitly the algorithm, that prove Theorem 1.

Let us focus on embedding the FFT graph $F(m) = (V_m, E_m)$ in the Hypercube $C(m + \lceil \log_2(m+1) \rceil)$. We specify the desired embedding by describing two labeling schemes.

- We assign each vertex $v \in V_m$ a unique d-bit label $L(v)$ (i.e., the labeling is injective).
- We assign each edge $(u, v) \in E_m$ a *bit-position* label $B(u, v) \in \{1, 2, \cdots, d\}$ such that $L(u)$ and $L(v)$ differ exactly in bit-position $B(u, v)$.

For each $i \in \{1, 2, \cdots, m\}$, there is a *bit-pair* (a_i, b_i) of bit-positions that are used for assignments to edges between levels $i-1$ and i of $F(m)$; i.e., all such butterfly edges "flip" the same pair of bits. In particular, each butterfly can be viewed as being labeled as indicated in Figure 1. We say that bit-position a *flips* on an edge of $F(m)$ if the labels of the endpoints of the edge differ in bit-position a.

The reader can verify easily that when we assign the d-bit label $L(v)$ to any single vertex v of $F(m)$, the labels of all remaining vertices can be specified uniquely by specifying

the *levelled bit-pair sequence* (*LBPS*) $S = (a_1, b_1), (a_2, b_2), \ldots, (a_m, b_m)$. The consistency of the labelling is proved by induction. We can, therefore, complete our task by labelling some input of $F(m)$ with the string $00\cdots0$ and using an appropriate LBPS to specify the labelling of the remaining vertices.

It remains to select an LBPS so that the labelling L is injective. To this end, call the LBPS $S = (a_1, b_1), (a_2, b_2), \ldots, (a_m, b_m)$ *proper* if, for each i, at least one of a_i or b_i does not occur in $\{a_j, b_j : j < i\}$, i.e., at an earlier level; call an a_i or b_i satisfying this condition *new*. We shall find a proper LBPS to effect the desired labelling.

Partition the levels $\{0, 1, \cdots, m\}$ of $F(m)$ into *tiers*: tier k is the set of levels $\{i : 2^k \leq i \leq 2^{k+1} - 1\} \cap \{0, 1, \cdots, m\}$. Let the singleton $\{0\}$ constitute tier -1. If the level of vertex v in $F(m)$ is in tier k, then we say that vertex v *is in* tier k.

Our embedding-labelling will be incremental: we shall obtain the labelling of $F(m+1)$ from the labelling of $F(m)$. Clearly, in the course of specifying our proper LBPS, a *new* bit-position can never lead to duplicated labels, so we shall always use any new bit-position as one becomes available (which happens when we must add a new dimension to the host Hypercube in order to accommodate the next bigger FFT graph). As we proceed from $F(m)$ to $F(m+1)$, the number of dimensions in the smallest Hypercube that will hold the FFT graph increases by at least one; hence, there is always at least one new bit-position to use in the labelling. Let us always call this new bit-position the a-position: at level i, it is bit-position a_i. Whenever $m = 2^k$ is a power of 2, then two new bit-positions are available for the expanded labelling when we proceed to $F(m+1)$. In this case, we call the pair (a_k, b_k) of new bit-positions the *shield positions* for tier k of all $F(n)$, $n > m$. It remains only to choose the b_i's that are not shield positions. We do this inductively, based on the tier number k. The base case $k = 1$ is trivial, as there are only shield positions. For purposes of induction, assume that the bit-positions for tier $k-1$ have been chosen. We choose the $2^k - 1$ b_i's for tier k as follows.

- For $1 \leq i \leq 2^{k-1} - 1$, we set $b_{2^k+i} = a_{2^{k-1}+i}$.
- For $i = 2^{k-1}$, we can choose either shield position from tier $k-1$; for definiteness, we set $b_{2^k+2^{k-1}} = a_{2^{k-1}}$.
- For $1 \leq i \leq 2^{k-1} - 1$, we set $b_{2^k+2^{k-1}+i} = b_{2^k+i}$.

Call the m-level version of the LBPS just described S_m.

We now must verify that S_m induces a unique label for each vertex of $F(m)$. We begin by simplifying our task, by showing that it suffices to look at S_m's behavior on a simple subgraph of $F(m)$. For any vertex $v \in V_m$, let $T(v)$ be the complete binary tree rooted at v and extending monotonically downward (i.e., to increasing levels) so that the leaves of $T(v)$ are outputs of $F(m)$. If vertices $u, v \in V_m$ are at the same level of $F(m)$, then there is a unique isomorphism $\iota_{u,v} : T(u) \to T(v)$ that preserves the bit-positions assigned to edges.

We now show that we need only specify an injective labelling of some $T(v)$ where v is an input of $F(m)$.

Lemma 1 *Let S be a proper LBPS for $F(m)$, and let $w \in V_m$ be any input of $F(m)$. Assign w the label $L(w) = 00\cdots0$. (As remarked earlier, the labels of remaining vertices are determined). If no two vertices of $T(w)$ are assigned the same label by the labelling of V_m induced by S, then no two vertices of $F(m)$ are assigned the same label.*

Proof. Let us suppose, for contradiction, that the Lemma is false. We are thus assuming that all vertices in the tree $T(w)$ rooted at input w are labelled uniquely, but that there are distinct vertices $u, v \in V_m$ for which $L(u) = L(v)$.

Note first that there is no input $z \in V_m$ of $F(m)$ such that both u and v reside in the tree $T(z)$. If there were such a z, then one sees easily that the vertices $\iota_{z,w}(u)$ and $\iota_{z,w}(v)$ of $T(w)$ must also be assigned the same label, since the labelling of $T(w)$ is dictated by the same LBPS S as is the labelling of $T(z)$. This would contradict the assumed uniqueness of labels in $T(w)$.

Let u reside at level ℓ_u in $F(m)$, and let v reside at level ℓ_v. Without loss of generality, say that $\ell_v \leq \ell_u$. Since u and v do not reside in the same $T(z)$, it follows that there exists an input z of $F(m)$ such that u is not in $T(z)$, but v is. Say we have chosen such a z.

Consider the graph $G(z, u) =_{def} T(z) \cup T(u)$. Choose a path P from u to v in $G(z, u)$, of the following form: P proceeds monotonically down $T(u)$ until a vertex s of $T(z)$ is reached; P then proceeds monotonically up $T(z)$ to an ancestor (not necessarily proper) of v; P finally goes down to v. The path P is guaranteed to exist, since the graph $G(z, u)$ is connected: the leaves of $T(z)$ comprise *all* of the outputs of $F(m)$ (z being an input of $F(m)$) while the leaves of $T(u)$ comprise some of the outputs of $F(m)$. Since $L(u) = L(v)$, it follows that every bit-position flips on edges of P an *even* number of times. Now, let $r \in V_m$ be the vertex that precedes s in P, and let $t \in V_m$ be the vertex that succeeds s in P; moreover, let k be the level of vertex s in $F(m)$ (easily, $\ell_v \leq \ell_u < k$). The edges (r, s) and (s, t) are the only edges in P that connect vertices on levels $k-1$ and k. Since these edges are distinct (t being in $T(z)$ while r is not) and share an endpoint, it follows that bit-position a_k flips on one of these edges, and bit-position b_k flips on the other. Since the LBPS S is proper, at least one of the bit-positions a_k, b_k must be new, hence flip only once in P, contradicting the even-flipping requirement. This contradiction establishes the Lemma. □

We shall now analyze the behavior of S_m on $F(m)$ by analyzing its behavior on one of the subtrees of $F(m)$ rooted at an input.

Lemma 2 *The LBPS S_m is proper. It does not repeat any label in a complete binary tree of height m.*

Proof. The LBPS S_m is proper by construction, so we can concentrate only on the uniqueness of labels.

Let w be an input of $F(m)$, so that $T(w)$ is a complete binary tree of height m. Assign w the label $00\cdots0$. Let us assume, to obtain a contradiction, that the distinct vertices u and v of $T(w)$ are assigned the same label under the LBPS S_m: $L(u) = L(v)$. Let u reside at level ℓ_u of $T(w)$, and let v reside at level ℓ_v. Say that ℓ_u is in tier k; then ℓ_v must also be in tier k, since the shield position that $L(u)$ has *on*[1] from tier k must also be *on* in $L(v)$. Without loss of generality, assume that ℓ_u (hence k) is smallest possible, and that $\ell_v \leq \ell_u$.

If $k < 2$, then we can check by inspection that the like-labeled vertices u and v cannot exist; hence, we assume henceforth that $k \geq 2$.

Let $P(u)$ (resp., $P(v)$) be the path from w to u (resp., to v) in $T(w)$. We simplify our task further, by showing that we can assume with no loss of generality that the path $P(v)$ has a special form. To this end, note that $P(u)$ (resp., $P(v)$) can be viewed as *choosing* either bit-position a_i or b_i at each level i, $1 \leq i \leq \ell_u$ (resp., $1 \leq i \leq \ell_v$). We say that a path *chooses the a-alternative* (resp., the *b-alternative*) at level i if it chooses bit-position a_i (resp., bit-position b_i). Say that, at a particular level i, we force path $P(u)$ (resp., path $P(v)$) to make the opposite choice of bit-position. Then the label of the vertex u' (resp., v') at level ℓ_u (resp., ℓ_v) that the new path leads to is $L(u') = L(u) \oplus 2^{a_i - 1} \oplus 2^{b_i - 1}$ (resp., $L(v') = L(v) \oplus 2^{a_i - 1} \oplus 2^{b_i - 1}$); hence, we obtain another pair of vertices with identical labels. It follows that by switching bit-position choices at precisely those levels i where $P(v)$ made the choice b_i, and by making the corresponding switches in $P(u)$, we obtain two vertices $u*$ and $v*$ with identical labels and with the property that the path $P(v*)$, which was obtained by switching all b_i's to a_i's in $P(v)$, always chooses the a-alternative. Thus, when we look at the like-labelled vertices u and v, we lose no generality by assuming that $P(v)$ chooses the a-alternative at every level.

We can now determine enough about the labels $L(u)$ and $L(v)$ to complete the proof of the Lemma. We begin by considering the choices that the paths $P(u)$ and $P(v)$ make as they traverse tier k. Since bit-positions $a_{2^k}, \ldots, a_{\ell_v}$ are new in tier k and are *on* in $L(v)$ (because $P(v)$ always chooses the a-alternative), they must be *on* in $L(u)$; hence, $P(u)$ makes the same choices as $P(v)$ at levels $2^k, \ldots, \ell_v$. Also, since bit-positions $a_{\ell_v+1}, \ldots, a_{\ell_u}$ are new in tier k and are *off* in $L(v)$ (since $P(v)$ terminates at v), path $P(u)$ must choose the b-alternative at levels $\ell_v + 1, \ldots, \ell_u$. We now know all the choices $P(u)$ and $P(v)$ make in tier k, so we turn our attention to tier $k - 1$.

Our analysis breaks into two cases, depending on which *half-tier* of tier k the vertices u and v reside in. Say the *first half-tier* of tier k comprises levels $2^k, 2^k + 1, \ldots, 2^k + 2^{k-1} - 1$, and the *second half-tier* comprises levels $2^k + 2^{k-1}, 2^k + 2^{k-1}, \cdots, 2^{k+1} - 1$.

We claim that u and v reside in the same half-tier. If this were not true then, by assumption, v would reside in the first half-tier, while u would reside in the second half-tier.

[1] Position i is *on* in a label if it contains a "1"; otherwise it is *off*.

Now, bit-position $a_{2^{k-1}}$ is *on* and bit-position $b_{2^{k-1}}$ is *off* in $L(v)$, since $P(v)$ always chooses the a-alternative, which is always new; hence, the same configuration must appear in $L(u)$. However, we have seen that $P(u)$ must always choose the b-alternative at levels below ℓ_v, so in particular, $P(u)$ must choose bit-position $b_{2^k+2^{k-1}} = a_{2^{k-1}}$. But now consider the choice that $P(u)$ made at level 2^{k-1}. If it chose the b-alternative at that level, then since that bit-position was new then and has not recurred, $L(u)$ would have *both* bit-positions $a_{2^{k-1}}$ and $b_{2^{k-1}}$ *on*. Alternatively, if $P(u)$ had chosen the a-alternative at that level, then this choice would have "cancelled" the choice of $b_{2^k+2^{k-1}} = a_{2^{k-1}}$, so $L(u)$ would have *both* bit-positions $a_{2^{k-1}}$ and $b_{2^{k-1}}$ *off*. Either contingency would contradict the assumption that $L(u) = L(v)$. We conclude that u and v are in the same half-tier.

Suppose that u and v are both in the first half-tier. Since bit-positions $a_{\ell_u-2^{k-1}+1}, \cdots, a_{2^k-1}$

- are new in tier $k-1$ (by definition),
- are *on* in $L(v)$ (since $P(v)$ always chooses the a-alternative),
- do not recur until levels $> \ell_u$,

these positions must be *on* in $L(u)$ also, so $P(u)$ must make the same choices as $P(v)$ at levels $\ell_u - 2^{k-1} + 1, \ldots, 2^k - 1$. By construction of S_m, bit-positions $b_{\ell_v+1}, \ldots, b_{l_u}$ are identical to bit-positions $a_{\ell_v-2^{k-1}+1}, \ldots, a_{\ell_u-2^{k-1}}$. These bit-positions are *on* in $L(v)$ by virtue of tier $k-1$ (where they are new); they must, therefore, be *on* in $L(u)$, since $L(u) = L(v)$. However, if they are *on* in $L(u)$, it must be by virtue of the b_i choices that we have already noted that $P(u)$ must make in tier k. It follows that $P(u)$ must choose the b-alternative at levels $\ell_v - 2^{k-1} + 1, \ldots, \ell_u - 2^{k-1}$: if $P(u)$ chose the a-alternative at any of those levels, that choice would combine with the matching b_i choice in tier k to turn off a bit-position that is *on* in $L(v)$.

We now have a total picture of the choices made by paths $P(u)$ and $P(v)$ in tier k and in the bottom portion of tier $k-1$, when u and v reside in the first half-tier. Let the path $P'(u)$ be obtained from $P(u)$ by truncating the latter at level $\ell_u - 2^{k-1}$; let the path $P'(v)$ be obtained from $P(v)$ by truncating the latter at level $\ell_v - 2^{k-1}$. Let $L'(u)$ and $L'(v)$ be the vertex-labels obtained by following $P'(u)$ and $P'(v)$, respectively. Since this truncation has removed the same bit-position settings from $P(u)$ and $P(v)$, one sees easily that $L'(u) = L'(v)$, even though the corresponding vertices reside within tier $k-1$ of $F(m)$. This contradicts the assumed minimality of k.

Finally, suppose that u and v are both in the second half-tier. By assumption, $P(v)$ chooses the a-alternative at levels $\ell_v - 2^k + 1, \ldots, 2^k - 1$. $P(u)$ must also make the same choices, since those bit-position settings in $L(u)$ must agree with $L(v)$; and we have seen that $P(u)$ does not choose those bit-positions when they recur (by definition of S_m) in the first half-tier of tier k.

We now have a total picture of the choices made by paths $P(u)$ and $P(v)$ in tier k and in the bottom portion of tier $k-1$, when u and v reside in the second half-tier. Let the path $P'(v)$ be obtained from $P(v)$ by truncating the latter at level $\ell_v - 2^{k-1}$; let $P'(u)$ be obtained from $P(u)$ by truncating the latter at level $\ell_u - 2^{k-1}$ and by choosing the b-alternative at levels $\ell_v - 2^{k-1}+1, \ldots, \ell_u - 2^{k-1}$. Let $L'(u)$ and $L'(v)$ be the vertex-labels obtained from following paths $P'(u)$ and $P'(v)$, respectively. One verifies as above that $L'(u) = L'(v)$ and that the corresponding vertices reside in tier $k-1$ of $F(m)$. Once again, we have contradicted the assumed minimality of k.

These contradictions establish Lemma 2. □

By Lemmas 1 and 2, we have established that the LBPS S_m produces a unique labeling of $F(m)$. Theorem 1 follows.

ACKNOWLEDGMENT. This research was supported in part by NSF Grant DCI-85-04308.

4. REFERENCES

1. A.V. Aho, J.E. Hopcroft, J.D. Ullman (1974): *The Design and Analysis of Computer Algorithms.* Addison-Wesley, Reading, MA.

2. F. Berman and L. Snyder (1984): On mapping parallel algorithms into parallel architectures. *Intl. Conf. on Parallel Processing.*

3. 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.

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

5. S.H. Bokhari (1981): On the mapping problem. *IEEE Trans. Comp., C-30.*

6. T.C. Chen, M.D.F. Schlag, C.K. Wong (1983): The hypercube connection network. IBM Report RC-10219.

7. L. Johnsson (1985): Basic linear algebra computations on hypercube architectures. Tech. Rpt., Yale Univ.

8. A.L. Rosenberg (1981): Issues in the study of graph embeddings. In *Graph-Theoretic Concepts in Computer Science: Proceedings of the International Workshop WG80,* Bad Honnef, Germany (H. Noltemeier, ed.) *Lecture Notes in Computer Science 100,* Springer-Verlag, New York 150-176.

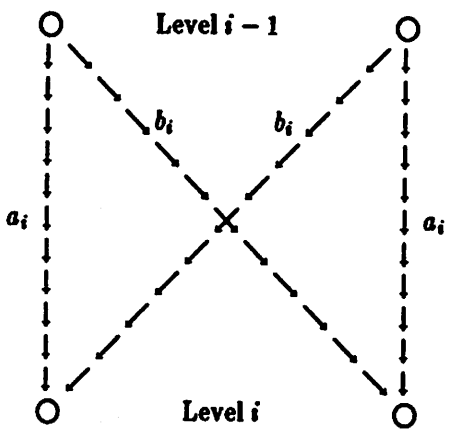

Figure 1. **A labelled butterfly.**