[12] T.R. Gross, J.L. Hennessy, S.A. Przybylski, and C. Rowen, "Measurement and evaluation of the MIPS architecture and processor," *ACM Trans. Comput. Syst.*, pp. 229–257, Aug. 1988.
[13] A. Vladimirescu and S. Liu, "The simulation of MOS integrated circuits using SPICE2," ERL Memo ERL M80/7, Electron. Res. Lab., Univ. of Cal., Berkeley, Oct. 1980.
[14] C. Tomovich, *MOSIS User Manual*, Rel. 3.1, USC/Inform. Sci. Inst., Marina del Rey, CA, 1988.

Embedding Binary X-Trees and Pyramids in Processor Arrays with Spanning Buses

Zicheng Guo and Rami G. Melhem

Abstract—**We study the problem of network embeddings in 2-D array architectures in which each row and column of processors are interconnected by a bus. These architectures are especially attractive if optical buses are used that allow simultaneous access by multiple processors through either wavelength division multiplexing or message pipelining, thus overcoming the bottlenecks caused by the exclusive access of buses. In particular, we define X-trees to include both binary X-trees and pyramids, and present two embeddings of X-trees into 2-D processor arrays with spanning buses. The first embedding has the property that all neighboring nodes in X-trees are mapped to the same bus in the target array, thus allowing any two neighbors in the embedded X-trees to communicate with each other in one routing step. The disadvantage of this embedding is its relatively high expansion cost. In contrast, the second embedding has an expansion cost approaching unity, but does not map all neighboring nodes in X-trees to the same bus. These embeddings allow all algorithms designed for binary trees, pyramids, as well as X-trees to be executed on the target arrays.**

Index Terms— **Alignment cost, embedding, spanning bus, pyramid, reflection index, X-tree**

I. Introduction and Problem Definition

Although the computational powers of parallel computers are potentially much larger than sequential ones, parallel computers suffer from inefficient interprocessor communications. Dealing with the communication inefficiency through either architecture or algorithm design, or both, has always been a key research issue in parallel computation. In order to improve the communication efficiency of parallel systems, buses, both electronic and optical, have been considered by many researchers for interconnecting processor arrays [1], [7], [9], [18], [19], [21]. In particular, 2-D processor arrays with broadcasting row and column buses have been suggested that allow for efficient solutions to various problems, including semigroup and prefix computations, image processing, computational geometry, and numerical computations [3], [16], [17]. Broadcasting buses, however, have low bandwidth because of their exclusive access mode of operation, and thus are not suited for tasks involving extensive interprocessor communications. This limitation may be alleviated if optical buses are used either with wavelength division multiplexing, which provides multiple channels on the same bus [7], or with space/time multiplexing, which allows message pipelining on a bus [9], [15]. In both cases, simultaneous bus access by multiple processors is permitted. As an illustration, in the following subsection, we briefly describe the principle of message pipelining on optical buses.

Manuscript received March 12, 1992; revised September 11, 1992. This work was supported by the U.S. Air Force under Grant AFOSR-89-0469, and by the National Science Foundation under Grant MIP-8901053.

Z. Guo is with the Department of Electrical Engineering, Louisiana Tech University, Ruston, LA 71272–0046, USA; e-mail: zg@engr.latech.edu.
R.G. Melhem is with the Department of Computer Science, University of Pittsburgh, Pittsburgh, PA 15261, USA.

IEEE Log Number 9401190.

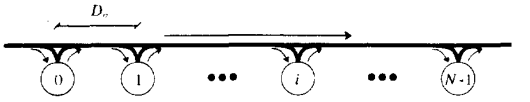

Fig. 1. A processor array connected with a spanning optical bus.

A. Message Pipelining on Optical Buses

Pipelined optical buses take advantage of two unique properties of optical signal transmissions in waveguides: unidirectional propagation and predictable path delays. Consider Fig. 1, which shows an array of N processors connected by an optical bus (waveguide). Each processor is coupled to the bus with two passive optical couplers, one for writing signals and the other for reading. Assume that the optical distance between each pair of adjacent processors is D_o and that a message consists of a sequence of b optical pulses, each having a width w in seconds. In contrast to the case of an electronic bus, where writing access to the bus is exclusive, all of the processors may write their messages on the optical bus simultaneously. This may be accomplished if all processors write their messages at the same instant and if the length of each message is smaller than D_o. Let c_g be the speed of light in the waveguide. Then we may ensure that messages sent by distinct processors do not collide with one another on the optical bus if $bwc_g < D_o$. Here, by "colliding," we mean that two optical signals injected on the bus by any two distinct processors arrive at some location on the bus simultaneously. With this collision-free condition satisfied, every processor can, in parallel, send a message to some other processor, and the messages will all travel from left to right on the bus in a pipelined fashion. To transfer messages in both directions in the same bus cycle, either a second waveguide can be added so that messages are transmitted in opposite directions on separate buses [9], or a folded waveguide can be used to allow messages to be delivered to processors on both sides of the source processor on the same bus [10].

Connecting N processors with a linear optical bus results in bus cycles of length $O(N)$. One solution to this problem is to extend the linear architecture of Fig. 1 to a 2-D architecture. Although the bus latency is significantly reduced to $O(\sqrt{N})$ in the 2-D architecture, in general it takes at least two steps, a row routing and a column routing, for two processors to communicate with each other. Such a two-step communication requires a message relay by an intermediate processor and involves an optical-electronic-optical information conversion in the middle of a message transfer. In the worst case, e.g., when all the messages from the same row are to be sent to the same column, up to $O(\sqrt{N})$ message relays by a single processor may be needed.

B. Alignment Cost

The problem studied in this short note is one of network embeddings in 2-D processor arrays with spanning buses, as shown in Fig. 2, where the processors in each row and each column are connected with a bus. A bus in Fig. 2 may be either optical or electronic and allows a processor to send a message to any other processor on the same bus in one routing step. It is clear from the above discussion that for efficient communications in such arrays, message relays should be minimized and preferably eliminated. This can be achieved by allocating communication processes to processors that share a common bus, thus avoiding the relay of messages.

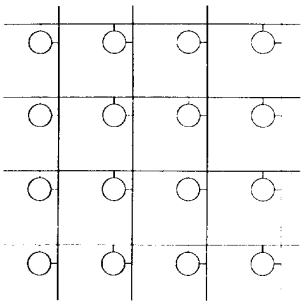

Fig. 2. A 2-D processor array with spanning buses.

Let $S = \{V_S, E_S\}$ be a process graph (called a source graph), where V_S is a set of processes and E_S is a set of edges between communicating processes. Let also T be the target network, which is a processor array with row and column spanning buses, onto which S is to be mapped. The quality of a mapping is often measured in terms of the *dilation cost*. Specifically, the dilation of an edge $\langle i, j\rangle \in E_S$ that is mapped to a path B in the target network, is $|B| - 1$, where $|B|$ is the number of nodes on B. The dilation cost for mapping the source graph into the target network is then defined as the maximum of all of the edge dilations. This measure, however, is of little value when the source graph is mapped into a 2-D processor array with spanning buses, because the bus connections allow two nodes to communicate in one step if the two nodes are mapped to the same bus, and in two steps otherwise. If two neighboring nodes i and j in the source graph are mapped to the same bus in the target array, we say that i and j are *aligned*. Clearly, it is desirable to obtain a mapping in which every pair of neighboring nodes in the source graph are aligned in the target array.

In order to measure how well neighboring nodes in a source graph S are aligned when mapped onto a target network with bus interconnections, we define a cost function, called *alignment cost*. Let $i \in V_S$ and $j \in V_S$ be two neighboring nodes connected by an undirected edge $\langle i, j\rangle \in E_S$, and let the cost of aligning i and j be ϕ_{ij}, where $\phi_{ij} = 0$ if i and j are mapped to the same bus in the target network, and $\phi_{ij} = 1$ otherwise. Because $\phi_{ij} = \phi_{ji} = 1$ when i and j are not mapped to the same bus, each misalignment will result in two cost units. The alignment cost, denoted Φ_a, of the mapping is defined as follows:

$$\Phi_a = \frac{\sum_{i,j} \phi_{i,j}}{|V_S|}, \qquad i, j \in V_S, \text{ and } \langle i, j\rangle \in E_S$$

Φ_a is the average number of misalignments per node of the source graph, and $0 \le \Phi_a \le \delta_S$ where δ_S is the degree of the source graph. In the above definition, $|V_S|$ is used to have the maximum value for Φ_a normalized to the degree of the source graph. This gives a maximum alignment cost independent of the size of the source graph, and makes it more convenient to evaluate embeddings of the same type of source graphs (e.g., X-trees) of different sizes. Note that the maximum alignment cost δ_S may result only when the degree of each node in the source graph is a constant. A mapping will be said to satisfy the *alignment condition* if its alignment cost is $\Phi_a = 0$, which is optimal. Another useful measure for evaluating graph embeddings is the *expansion cost*, denoted Φ_e, which is defined as $\Phi_e = |V_T|/|V_S|$, where $|V_T|$ is the number of nodes in the target network. These costs will be used to evaluate our embeddings of X-trees as defined in the next subsection.

C. X-Trees

Consider an L-level tree in which each node has C children, where $C = 2$ or 4. The tree is augmented with interconnections among the nodes at the same level l. $0 \le l < L$. The first class of X-trees is the X-binary-trees [6], where $C = 2$ and there are 2^l nodes on each level l. Assume that the nodes on each level are numbered linearly from left to right, with the leftmost node having an index 0. Then, on level l of an X-binary-tree, each node y, $0 \le y < 2^l$ is connected to node $(y + 1) \bmod 2^l$. (See Fig. 3(a), which is simplified by not drawing the wraparound connection completely on each level.)

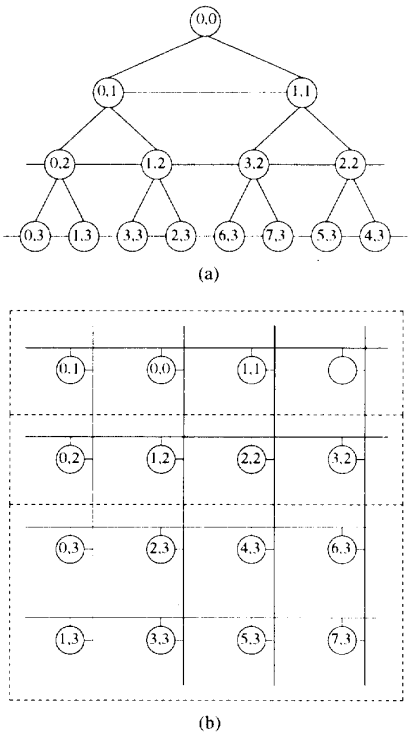

Fig. 3. (a) An X-binary-tree, with the two numbers (r, l) in each node indicating the reflection index and the level number, respectively, of the node. (b) Its embedding, M_2, in a 2-D array with spanning buses.

In the second class of X-trees that we consider, $C = 4$, and there are 4^l nodes on each level l. These nodes are interconnected as a $2^l \times 2^l$ mesh with wraparound connections. Specifically, let $(x, y), 0 \le x, y < 2^l$ be the row/column position of a node in the $2^l \times 2^l$ mesh representing level l of the X-tree. Then the wraparound connections connect node $(0, y)$ to $(2^l - 1, y)$, and node $(x, 0)$ to $(x, 2^l - 1)$ (See Fig. 4(a), where wraparound connections are omitted for clarity.) The resulting architecture will be called an X-quad-tree, which is an extension of the X-binary-tree. Thus, an X-binary-tree is a binary tree with each level connected as a ring, and an X-quad-tree is a quad tree with each level connected as a torus. The term X-tree is used hereafter to refer to both X-binary-tree and X-quad-tree.

In an L-level X-tree, each node has $2C + 1$ neighbors, except the root (at level 0), which has only C children, and the leaf nodes (at level $L - 1$) each of which has C neighbors at the same level and a parent at the level above. When embedding an X-tree into a 2-D array with spanning buses, all of the neighboring nodes in the X-tree must be mapped to either the same row or the same column of the target array, so that the alignment condition can be satisfied. To faciliate future discussions, we define the alignment condition for the embedding of an L-level X-tree in a 2-D array as follows.

H1) Each node on level $l, 1 \le l \le L - 1$, of the source X-tree and its neighbors on the same level are mapped to either the

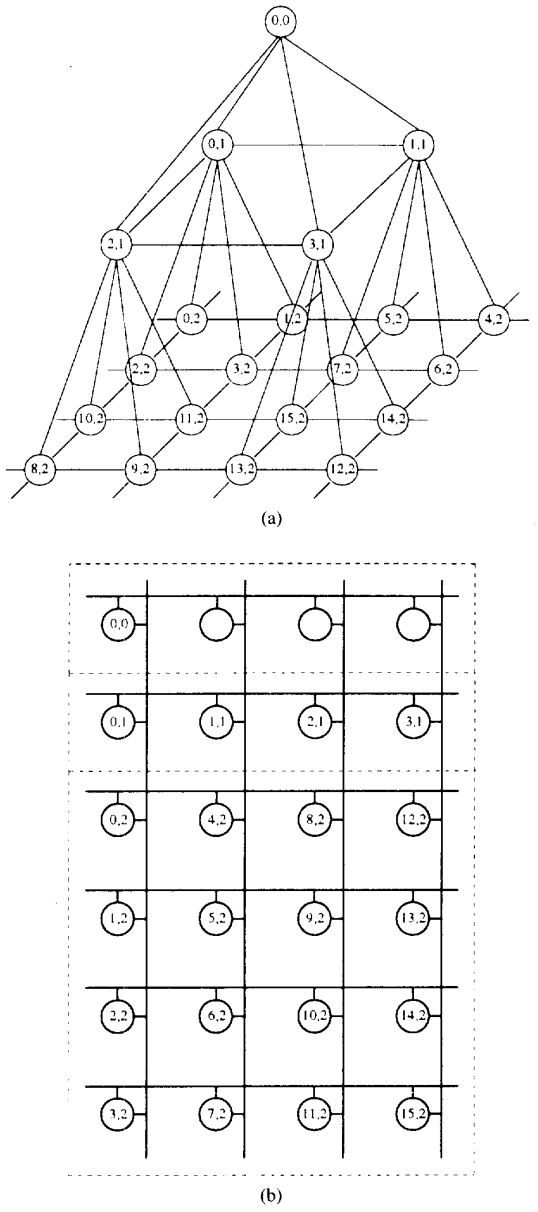

Fig. 4. (a) An X-quad-tree, with the two numbers (r, l) in each node indicating the reflection index and the level number, respectively, of the node. (b) Its embedding, M_2, in a 2-D array with spanning buses.

same row or the same column in the target 2-D array with spanning buses.

H2) Each parent node on level $l, 0 \leq l \leq L-2$ of the source X-tree and its children on level $l+1$ are mapped to either the same row or the same column in the target 2-D array with spanning buses.

Note that once an embedding of X-trees into a 2-D array is obtained, all algorithms designed for binary trees, pyramids as well as X-trees, can be run on the target array. In the case of running a binary tree algorithm, all connections at the same level are ignored. In the case of running a pyramid algorithm, the wraparound connections at each level are not used. Numerous applications will benefit from such embeddings because many efficient parallel algorithms have been designed for binary trees, X-binary-trees, and pyramids. Examples include selection, scheduling, sorting, prefix algorithms, matrix operations, graph theoretic problems, and various image and vision computations [2], [4]–[6], [12], [14].

In Section II, we define, for meshes, an indexing scheme, called *the reflection index*, and a coding scheme called *the reflection code*. These schemes will be used to specify the X-tree embeddings, because each level l of an X-tree is a $(C/2)^l \times 2^l$ mesh (with wraparound connections). The reflection code was first introduced in [8] for $2^i \times 2^i$ square meshes and used to obtain an embedding of pyramids in 2-D arrays with pipelined optical buses. The definition of the reflection code in this short note extends that of [8] to $2^i \times 2^j$ meshes for arbitrary integers i and j. This allows us to embed binary trees, X-trees, and pyramids using the reflection code. This result is presented in Section III as our first embedding, which is proved to achieve an optimal alignment cost at the expense of a relatively high expansion cost. The second embedding, presented in Section IV, is defined using the reflection index, and achieves an optimal expansion cost. The alignment cost of the second embedding, however, is not optimal. Finally, these results are summarized in Section V.

II. Reflection Index and Reflection Code

First, the reflection index is defined. Consider a mesh of size $2^i \times 2^j$ for arbitrary integers $i \geq 0$ and $j \geq 0$. Let (x, y) be the node at row x and column $y, 0 \leq x < 2^i$ and $0 \leq y < 2^j$. The reflection index for (x, y) is defined by two transforms, G and R, on (x, y). The first transform, G, results in a Gray code [13], to which the second transform, R, is applied to obtain the reflection index. Let a be an integer with binary representation $a_{k-1} \cdots a_0$. Then the transform G is defined as follows:

$$G(a_{k-1}a_{k-2}\cdots a_1a_0) = z_{k-1}z_{k_2}\cdots z_1z_0, \tag{1}$$

where $z_{k-1} = a_{k-1}, z_h = a_{h+1} \oplus a_h$ for $h = 0, \cdots, k-2,$, and $\oplus$ is the logical Exclusive-OR operator. The Gray code, denoted (e, f), for (x, y) is then given by the following equation:

$$(e, f) = (G(x), G(y)). \tag{2}$$

Let $e_{i-1}\cdots e_0$ and $f_{j-1}\cdots f_0$ be the binary representations of e and f, respectively. The Gray code index for (x, y) is defined as $g = ef = e_{i-1}\cdots e_0 f_{j-1}\cdots f_0$.

To define the reflection index, the notion of binary numbers with null bits is now introduced. The notion is best illustrated by an example. Let a be an integer with binary representation $a_3a_2a_1a_0$, where a_3, a_1 and a_0 are ordinary bits and a_2 is a null bit, denoted by $*$. Then the actual value of integer a is obtained by throwing away the null bit; that is, $a = a_3 * a_1a_0 = a_3a_1a_0$. For example, if $a_3 = a_0 = 1$ and $a_1 = 0$, then $a = 1^*01 = 101 = 5$.

For notational convenience in defining the reflection index, e or f is augmented with $|i-j|$ null bits at higher bit positions, so that e and f have the same number of bits in their binary representations. Specifically, if $i < j$, the binary representation of e is augmented with $j-i$ null bits, so that one can write $e = e_{j-1}\cdots e_i e_{i-1}\cdots e_0$, where e_{j-1} through e_i are null bits. If $i \geq j$, however, f is augmented with $i-j$ null bits. If $k = \max\{i, j\}$, then the reflection index r for (x, y) in a $2^i \times 2^j$ mesh is obtained by shuffling the bits of g, which is the Gray code index of (x, y). That is,

$$\begin{aligned} r = R(g) &= R(ef) = R(G(x)G(y)) \\ &= e_{k-1}f_{k-1}e_{k-2}f_{k-2}\cdots e_0f_0. \end{aligned} \tag{3}$$

It can be checked that after throwing away the null bits, the number of bits in r is $i+j$, resulting in a binary representation $r = r_{i+j-1}\cdots r_0$.

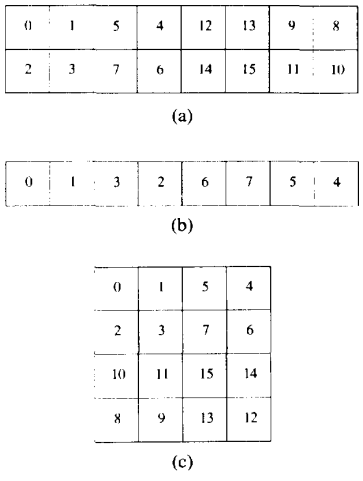

Fig. 5. The reflection index. (a) 2×8 mesh. (b) Linear array. (c) Square mesh.

An example of the reflection index is shown in Fig. 5(a) for a 2×8 mesh. Two special cases of the mesh size are important to us. When $i = 0$ (or $j = 0$), the mesh degenerated to a linear array, the reflection index is the same as the Gray code index. As the other special case, when $i = j$, the mesh being a square, the reflection index is obtained by shuffling the bits (without null bits) of the Gray code index. Fig. 5(b) and 5(c) show the reflection indices for a linear array and a square mesh, respectively. The reflection indices (and the reflection codes, defined later) for these two special mesh sizes will facilitate our definitions of X-tree embeddings, because each level of an X-tree is a linear array with wraparound connections in the case of an X-binary-tree, and a square mesh with wraparound connections in the case of an X-quad-tree.

The reflection index for a mesh of size $2^i \times 2^j$, can be obtained recursively by successive reflections, thus the name *reflection index*. Specifically, starting with a single node 0, first perform alternate column and row reflections until all the 2^i rows are exhausted (assuming $i < j$, the case where $i \geq j$ is similar), and then do only column reflections until all of the 2^j columns are exhausted. The reflection index has the properties of adjacency and squareness, as established in Lemmas 1 and 2 below.

Lemma 1: (Adjacency of the reflection index) The reflection indices for two adjacent nodes (x, y) and $(x, (y + 1) \bmod 2^j)$ or $((x + 1) \bmod 2^i, y)$ in a $2^i \times 2^j$ mesh are at Hamming distance 1.

It can be proved that $G(x)G(y)$ gives the Gray code index g for (x, y) [11]. Since, from (3), the reflection index r is a bit permutation over g, its adjacency property carries over.

Lemma 2: (Squareness of the reflection index) If a $2^i \times 2^j$ mesh is partitioned into square blocks, each of size $2^h \times 2^h$, then the 2^{2h} nodes in each block have consecutive reflection indices.

Proof: Let (x, y) be the node at row x and column $y, 0 \leq x < 2^i$ and $0 \leq y < 2^j$, in a $2^i \times 2^j$ mesh, and let x and y have binary representations $x_{i-1} \cdots x_h x_{h-1} \cdots x_0$ and $y_{j-1} \cdots y_h y_{h-1} \cdots y_0$, respectively. Then the 2^{2h} nodes in each $2^h \times 2^h$ block have the same higher $i - h$ bits $x_{i-1} \cdots x_h$ in x and $j - h$ bits $y_{j-1} \cdots y_h$ in y, but different lower h bits $x_{h-1} \cdots x_0$ and $y_{h-1} \cdots y_0$ in x and y, respectively. Let the reflection indices for these nodes be $r = r_{i+j-1} \cdots r_{2h} r_{2h-1} \cdots r_0$. To prove the squareness, we need show only that bits $r_{i+j-1} \cdots r_{2h}$ depend only on $x_{i-1} \cdots x_h$ and $y_{j-1} \cdots y_h$, and are thus the same for the 2^{2h} nodes in each $2^h \times 2^h$ block. It can be shown from (1) and (2), however, that $e_{i-1} \cdots e_h$ and $f_{j-1} \cdots f_h$, which are the higher bits in e and f of the Gray code (e, f) for (x, y), depend only on $x_{i-1} \cdots x_h$ and $y_{j-1} \cdots y_h$, respectively. Thus, to prove the squareness, it is sufficient to show that bits $r_{i+j-1} \cdots r_{2h}$ depend only on $e_{i-1} \cdots f_h$ and $f_{j-1} \cdots f_h$. This can be easily shown to be true from (3). □

Because the reflection index will be used to define one of our X-tree embeddings, we are particularly interested in knowing what are the reflection indices of the C children on level l of an X-tree that share the same parent. Note that these nodes are in blocks of size $(C/2) \times 2$ in a $(C/2)^l \times 2^l$ mesh corresponding to level l of the X-tree. From Lemma 2, we have Corollary 1, which follows.

Corollary 1: Let $\hat{r}$ be the reflection index for any of the C nodes in each $(C/2) \times 2$ block of a $(C/2)^l \times 2^l$ mesh, where $C = 2$ or 4. Then we have:

$$\hat{r} = a e_0 f_0, \tag{4}$$

where $a = \hat{r}_{t-1} \cdots \hat{r}_2$, and $t = Cl/2$ is the number of bits in $\hat{r}$.

Corollary 1 tells us that the binary representations of the reflection indices for the C children of any node in an X-tree have the same higher significant bits $\hat{r}_{t-1} \cdots \hat{r}_2$, and are different only in the two lowest bits $\hat{r}_1 = e_0$ and $\hat{r}_0 = f_0$, where e_0 and f_0 are the least significant bits in the Gray code (e, f) of each node, as defined in (2). Note that e, and thus e_0, is null if $C = 2$.

The adjacency and squareness properties of the reflection index can be checked from the examples in Fig. 5. It is interesting to compare the reflection index with two well-known indexing schemes for meshes: the Gray code index and the shuffled row major index [20]. Like the reflection index, the Gray code index also possesses adjacency, but it does not have squareness (except when 2-D meshes degenerate into linear arrays). On the contrary, the shuffled row major index also has squareness; it does not possess adjacency, however. Thus, neither the Gray code index nor the shuffled row major index has both adjacency and squareness properties, which, as will be seen, are both crucial to our X-tree embeddings.

Next the reflection code is defined. The reflection code, denoted (p, q), for (x, y) in a $2^i \times 2^j$ mesh, is also defined by two transforms. The first transform, G, is defined the same as before, which gives us the Gray code (e, f) for (x, y). Then, applying the second transform, R', to (e, f), one obtains the reflection code (p, q). R' is defined in (5) below. Note that e or f has been augmented with null bits at higher bit positions to make the number of bits in both e and f equal to k, where $k = \max\{i, j\}$ as defined before. Thus, given a node (x, y) in a mesh of size $2^i \times 2^j$, its reflection code (p, q) is defined as follows:

$$(p, q) = R'(G(x), G(y)). \tag{6}$$

Fig. 6 shows examples of the reflection code. Comparing (5) with (3), it is seen that the two numbers p and q of the reflection code (p, q) can be obtained by applying R to the odd and even bits, respectively, in the binary representations of e and f of the Gray code (e, f). Specifically, assume for convenience that k is even, and let $\dot{e} = e_{k-1} e_{k-3} \cdots e_1$ and $\dot{f} = f_{k-1} f_{k-3} \cdots f_1$ (if k is odd, then $\dot{e} = e_{k-2} e_{k-4} \cdots e_1$ and $\dot{f} = f_{k-2} f_{k-4} \cdots f_1$). That is, $\dot{e}$ and $\dot{f}$ consist of only the odd bits in the binary representation of e and f, respectively. Similarly, let $\ddot{e}$ and $\ddot{f}$ consist of only the even bits in

$$(p, q) = R'(e, f) = \begin{cases} (e_{k-1} f_{k-1} e_{k-3} f_{k-3} \cdots e_1 f_1, e_{k-2} f_{k-2} e_{k-4} f_{k-4} \cdots e_0 f_0), & k \text{ even}, \\ (e_{k-2} f_{k-2} e_{k-4} f_{k-4} \cdots e_1 f_1, e_{k-1} f_{k-1} e_{k-3} f_{k-3} \cdots e_0 f_0), & k \text{ odd}. \end{cases} \tag{5}$$

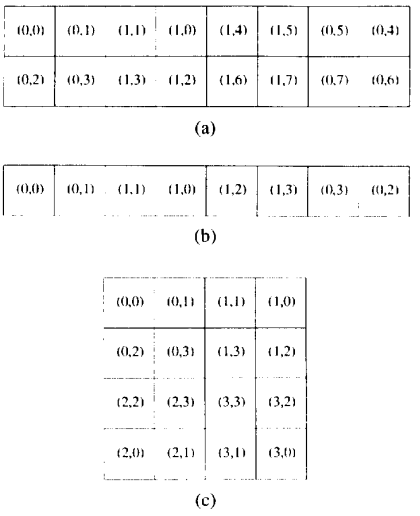

Fig. 6. The reflection code. (a) 2×8 mesh. (b) Linear array. (c) Square mesh.

the binary representations of e and f, respectively. Then $p = R(\dot{e}\dot{f})$ and $q = R(\ddot{e}\ddot{f})$. Therefore, like the reflection index, the reflection code for a mesh of size $2^i \times 2^j$ can also be obtained by successive reflections. Because the reflection code is a 2-tuple (p, q), however, one needs to do reflections on the two numbers p and q alternatively. Further, the reflection code also has the properties of adjacency and squareness. Lemmas 3 and 4 below are similar to Lemmas 1 and 2, respectively.

Lemma 3: (Adjacency of the reflection code) The reflection codes for two adjacent nodes (x, y) and $(x, (y + 1) \bmod 2^j)$ or $((x + 1) \bmod 2^i, y)$ in a $2^i \times 2^j$ mesh are at Hamming distance 1.

Lemma 4: (Squareness of the reflection code) If a $2^i \times 2^j$ mesh is partitioned into square blocks, each of size $2^h \times 2^h$, then the reflection codes (p, q) for the 2^{2h} nodes in each block have consecutive values of p and q.

Corollary 2: Any of the C nodes in each $(C/2) \times 2$ block of a $(C/2)^l \times 2^l$ mesh, where $C = 2$ or 4, has the following reflection code:

$$(P, Q) = (P, AE_0F_0), \tag{7}$$

where $A = Q_{t-1} \cdots Q_2$, and $t = \lceil Cl/4 \rceil$ is the number of bits in Q.

It is clear from Corollary 2 that the first variable, P, in the reflection codes for the C children of any node in an X-tree has the same value. The second variable, Q, in their reflection codes is different only in the two lowest bits $Q_1 = E_0$ and $Q_0 = F_0$, where QE_0 and F_0 are the least significant bits in the Gray code (E, F) of each node, and E is null for $C = 2$.

The reflection code and the reflection index defined above allow us to define two embeddings, presented in the next two sections, with optimal alignment cost and optimal expansion cost, respectively.

III. An X-Tree Embedding With Optimal Alignment Cost

Consider an X-tree of L levels, with the root at level 0. Level $l, 0 \le l < L$, of the X-tree can be viewed as a mesh of size $(C/2)^l \times 2^l$. To each node at level l, assign a label (x, y, l), where $(x, y), 0 \le x < (C/2)^l$ and $0 \le y < 2^l$ is the row and column position of a node in the mesh corresponding to level l. As a result, the C children of $(x, y, l), 0 \le l \le L - 2$ are $(x_{l-1} \cdots x_0 X_0, y_{l-1} \cdots y_0 Y_0, l + 1) = (xX_0, yY_0, l + 1)$, where X_0 and Y_0 are either 0 or 1 if $C = 4$, and x and X_0 are null if $C = 2$. It should be clear that the C children can be identified by the binary number X_0Y_0 for different values of X_0 and Y_0. The C neighbors of $(x, y, l), 1 \le l \le L - 1$ on the same level are $((x \pm 1) \bmod (C/2)^l, (y \pm 1) \bmod 2^l, l)$. Because each node knows the labels of its neighbors and the Gray code and the reflection code for any node can be determined by using (2) and (6), respectively, the Gray code and the reflection code for each neighbor of (x, y, l) are also known to (x, y, l). For convenience, one may also refer to node (x, y, l) by (e, f, l), (p, q, l) or (r, l), where $(e, f), (p, q)$, and r are the Gray code, the reflection code, and the reflection index of (x, y), respectively. In the case where the level number l in the label of a node is not of interest, one may omit l and simply write $(x, y), (e, f), (p, q)$, or r.

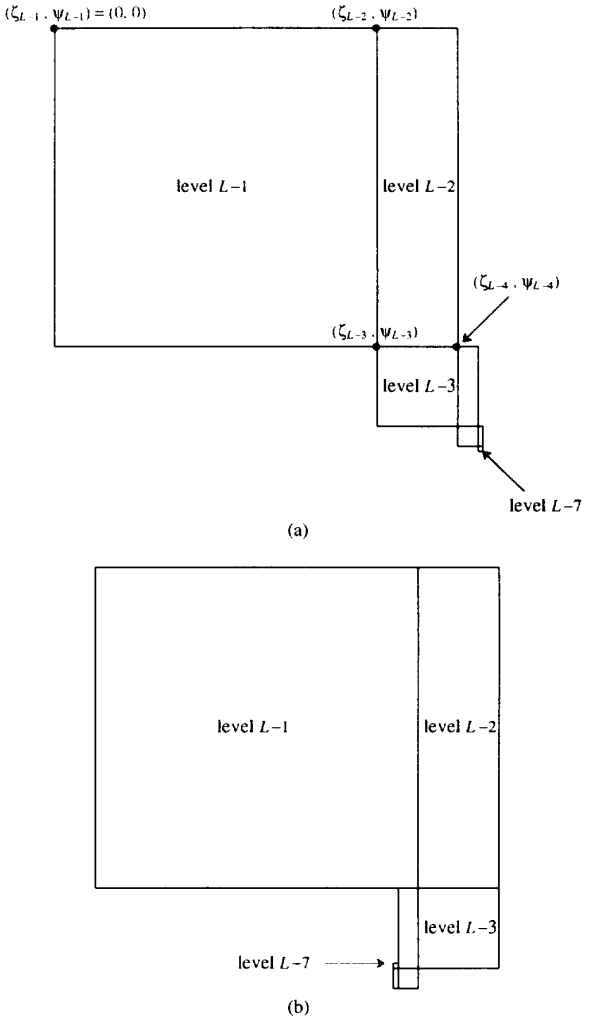

Fig. 7. (a) Embedding M_1 of an L-level X-tree, L odd, obtained by embedding each level of the X-tree into a square or rectangular area in a 2-D array with spanning buses. (b) Levels $L - 4$ through 0 are flipped over to reduce the expansion cost without affecting the alignment cost.

The embedding of an L-level X-tree into a 2-D array with spanning buses (the target array) can be obtained by embedding each level l (a source mesh), $0 \le l < L$, of the X-tree into a square or rectangular area (a target subarray) in the target array. Specifically, if l is even, then the target subarray is of size $C^{l/2} \times C^{l/2}$; if l is odd, however, then the target subarray is of size $C^{(l+1)/2} \times C^{(l-1)/2}$ (see Fig. 7(a)). The nodes in each level are mapped to a target subarray so

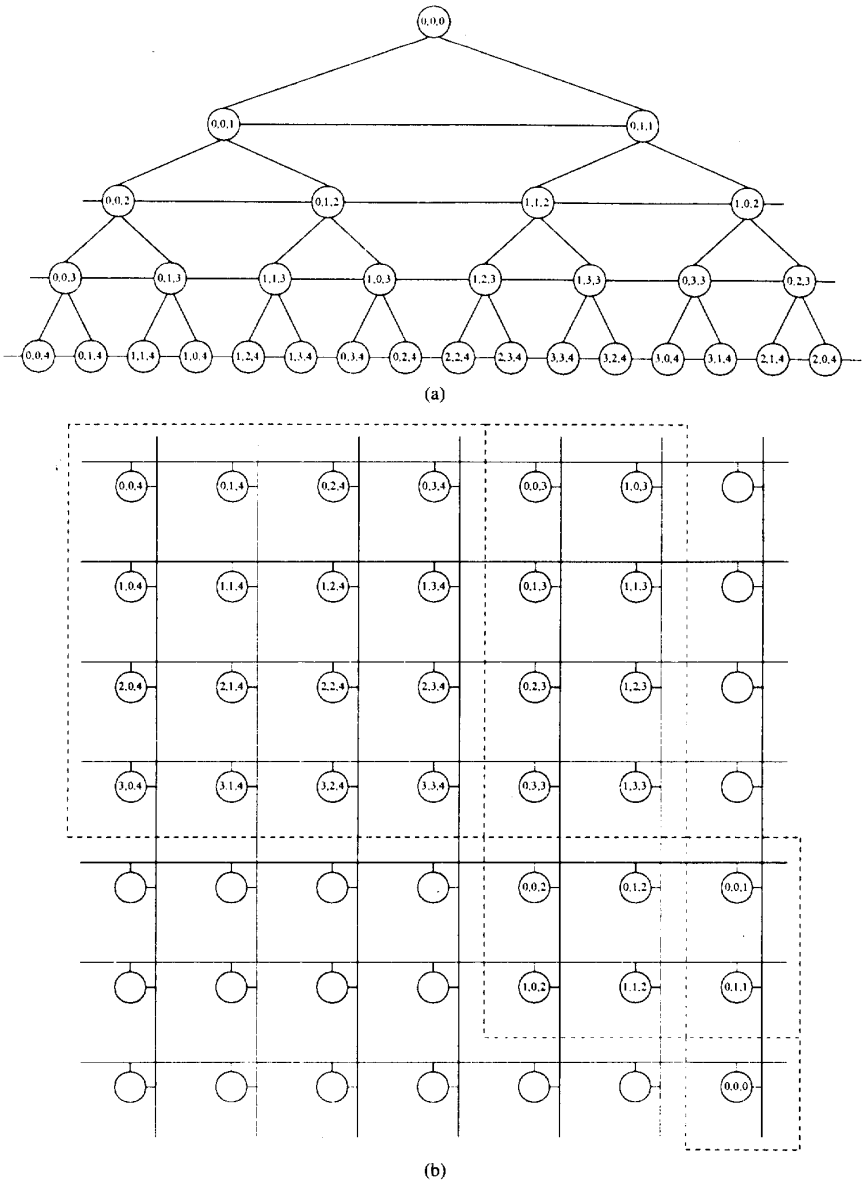

Fig. 8. (a) An X-binary-tree with the three numbers (p, q, l) in each node indicating its reflection code (p, q) and level number l, respectively. (b) Its embedding, M_1, in a 2-D array with spanning buses.

that condition H1, defined in Section I, is satisfied. The L resulting target subarrays are then properly positioned to form the target array, so that condition H2 is satisfied. For odd L, the resulting target array is square, and for even L, the resulting array is rectangular. In the remainder of this section, we simplify our formulas by assuming that L is odd. The formulas for the case of even L are slightly different and may be obtained in a similar manner. Specifically, our first embedding is defined by a mapping M_1, which maps each node (p, q, l) of the X-tree to a node $M_1(p, q, l)$ in the target array. The mapping M_1 is defined as follows:

$$M_1(p,q,l) = \begin{cases} (p + \zeta_l, q + \Psi_l), & \text{for } l \text{ even}, \\ (q + \zeta_l, p + \Psi_l), & \text{for } l \text{ odd}, \end{cases} \tag{8}$$

where $\Psi_{L-1} = 0$, $\Psi_l = \Psi_{l+1} + C^{(l+1)/2}$ if l is odd, and $\Psi_l = \Psi_{l+1}$ if l is even; and $\zeta_{L-1} = 0$, $\zeta_l = \zeta_{l+1}$ if l is odd and $\zeta_l = \zeta_{l+1} + C^{(l+2)/2}$ if l is even. That is, a node (p, q) on level l of the X-tree is mapped to position (p, q) in a $C^{l/2} \times C^{l/2}$ mesh if l is even, or (q, p) in a $C^{(l+1)/2} \times C^{(l-1)/2}$ mesh if l is odd. The origin of this mesh is then shifted to position (ζ_l, Ψ_l) of the target array. Figs. 8(b) and 9(b) show the embeddings for the X-trees in Figs. 8(a) and 9(a), respectively, into a 2-D array with spanning buses. Dashed boxes in Figs. 8(b) and 9(b) indicate the embeddings of individual levels (compare with Fig. 7(a), which is drawn to the scale of X-quad-trees where the area of each successive level is quadrupled). In order to prove that the embedding M_1 satisfies the alignment condition, we start by proving the following lemma.

Lemma 5: Let $s = E_0F_0$, where E_0 and F_0 are either 0 or 1 if $C = 4$, and E_0 is a null bit if $C = 2$. Then, in an L-

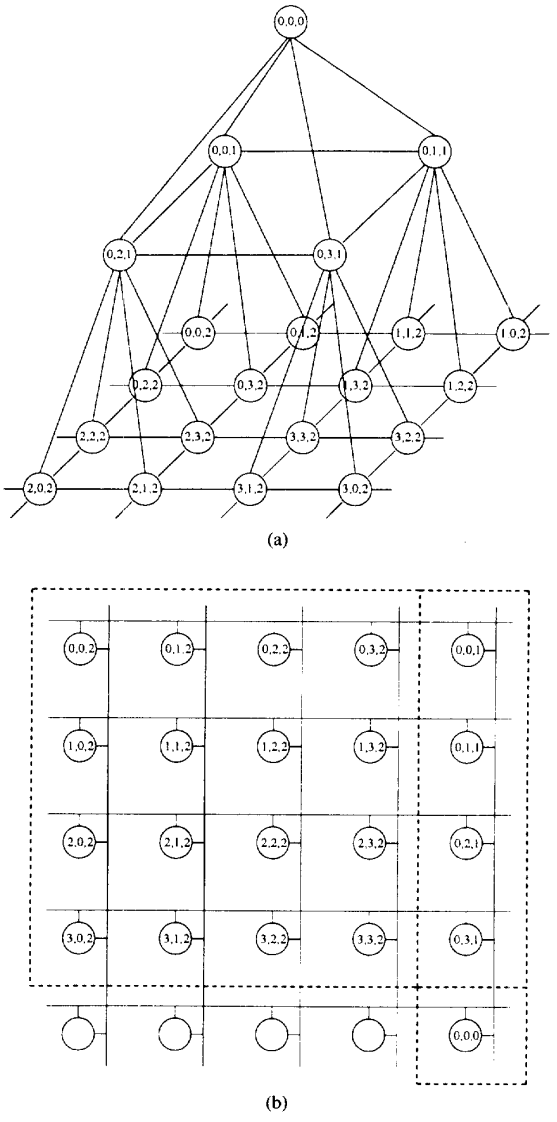

Fig. 9. (a) An X-quad-tree with the three numbers (p, q, l) in each node indicating its reflection code (p, q) and level number l, respectively. (b) Its embedding, M_1, in a 2-D array with spanning buses.

level X-tree, the C children of (p, q, l) are $(q, ps, l + 1)$ where $0 \leq l \leq L - 2$.

Proof: Note that if (e, f, l) is the parent of $(E, F, l+1)$, then:

$$(E, F) = (E_l \cdots E_1 E_0, F_l \cdots F_1 F_0)$$
$$= (e_{l-1} \cdots e_0 E_0, f_{l-1} \cdots f_0 F_0) = (eE_0, fF_0).$$

Let (p, q) and (P, Q) be the reflection codes for (e, f) and (E, F), respectively. Assuming that l is even (the case for odd l follows similarly), then, from (5), we have:

$$(p, q) = (e_{l-1} f_{l-1} e_{l-3} f_{l-3} \cdots e_1 f_1, e_{l-2} f_{l-2} e_{l-4} f_{l-4} \cdots e_0 f_0),$$

and

$$(P, Q) =$$
$$(E_{l-1} F_{l-1} E_{l-3} F_{l-3} \cdots E_1 F_1, E_l F_l E_{l-2} F_{l-2} \cdots E_0 F_0) =$$
$$(e_{l-2} f_{l-2} e_{l-4} f_{l-4} \cdots e_0 f_0, e_{l-1} f_{l-1} e_{l-3} f_{l-3} \cdots e_1 f_1 E_0 F_0) =$$
$$(q, ps). \quad (9)$$

Note that if $C = 2$, all the bits of E, including E_0 are null bits. That is, $s = F_0$ for $C = 2$. This completes the proof of Lemma 5. □

The interested reader may want to compare (9), noting that $s = E_0 F_0$, with (7) to see the effect of squareness on obtaining Lemma 5, which, together with adjacency, is crucial to the proof of the following theorem.

Theorem 1: The X-tree embedding M_1, defined in (8), satisfies the alignment conditions H1 and H2.

Proof: To prove H1, notice that from Lemma 3, all the neighboring nodes of (p, q, l) on level l are at Hamming distance 1 from (p, q, l). Thus, it is sufficient to show that all nodes at Hamming distance 1 from (p, q, l) on level l are mapped to the same row or column as (p, q, l). Consider two nodes (p, q, l) and $(p, \overline{q}, l)$ (or (p, q, l) and $(\overline{p}, q, l)$), where q and $\overline{q}$ (or p and $\overline{p}$) differ by exactly one bit in their binary representations. From (8), if l is even, the two nodes are mapped to the same row $p + \zeta_l$ (or to the same column $q + \Psi_l$), and if l is odd, the two nodes are mapped to the same column $p + \Psi_l$ (or the same row $q + \zeta_l$), thus proving H1.

To prove H2, note that from Lemma 5, node (p, q, l) has C children $(P, Q, l + 1) = (q, ps, l + 1)$. When l is even, according to (8), the parent is mapped to $(p + \zeta_l, q + \Psi_l)$, and noting that $l + 1$ is odd, the children are mapped to $(Q + \zeta_{l+1}, P + \Psi_{l+1}) = (ps + \zeta_{l+1}, q + \Psi_{l+1})$. Because $\Psi_l = \Psi_{l+1}$ for even l, then $q + \Psi_l = q + \Psi_{l+1}$. Thus, the parent (p, q, l) and its C children $(P, Q, l + 1)$ are mapped to the same column.

When l is odd, from (8), the parent (p, q, l) is mapped to $(q + \zeta_l, p + \Psi_l)$, and the children are mapped to $(P + \zeta_{l+1}, Q + \Psi_{l+1}) = (q + \zeta_{l+1}, ps + \Psi_{l+1})$. Because $\zeta_l = \zeta_{l+1}$ for odd l, then $q + \zeta_l = q + \zeta_{l+1}$. Thus, the parent (p, q, l) and its children $(P, Q, l + 1)$ are mapped to the same row. Therefore, H2 is satisfied by embedding M_1, completing the proof of Theorem 1. □

Note that because each node knows the reflection code of its neighbors (see the discussion at the beginning of this section), it is clear from the proof of Theorem 1 that each node knows whether a neighbor is mapped to the same row or to the same column. For example, if two neighboring nodes on the same level l have the same p value in their reflection codes, then the two nodes will be mapped to the same row if l is even, or to the same column if l is odd.

An important issue in embeddings is how efficiently the routing in the source architecture can be emulated in the target architecture, and what the incurred overhead is. To see this, let us assume that in one routing step, every node in the source X-tree sends a message to one of its neighbors. In the case of optical buses, because all nodes on a bus can write their messages on the bus simultaneously and all neighboring nodes in the source X-tree are aligned, each routing step in the source X-tree can be emulated in one step in the target network. Thus, the emulation incurs no overhead. Note that if two neighboring nodes in the source X-tree are not aligned, then each message transfer between the two misaligned nodes requires a message relay by an intermediate node. As discussed in Section I, up to $O(\sqrt{N})$ messages may have to be relayed by the same node, which subsequently increases the number of steps that are required to emulate one X-tree routing step. Therefore, aligning neighboring nodes in the X-tree significantly improves the efficiency of the emulation task.

In the case of exclusive access buses, only one distinct message can be written on a bus at any step. Thus, more than one step in the target array will be required to emulate one routing step in the X-tree. The number of steps taken in order to emulate one X-tree routing step will, in the worst case, be equal to the maximum number, N_M, of messages to be transmitted on the same bus in the target array. To find N_M, we consider three types of routing steps in the X-tree.

1) Each child node sends one message to its parent.
2) Each parent node broadcasts the same message to all of its children or sends one message to one of its children.
3) Each node sends one message to any neighbor on the same level.

These are the most commonly used routing patterns in X-tree algorithms. Let N_a, N_b, and N_c be the maximum number of messages to be transmitted on the same bus for the three types of routing, respectively. Then $N_M = \max\{N_a, N_b, N_c\}$. Clearly, in type 1 routing, a node may receive C messages, and in types 2 and 3 routing, each node receives at most one message. It can be shown that $N_a = C^{(L-1)/2}$, $N_b = C^{(L-3)/2}$, and $N_c = C^{(L-1)/2} + C^{(L-3)/2}$. Therefore, $N_M = N_c = C^{(L-1)/2} + C^{(L-3)/2}$, which is equal to the maximum number of nodes mapped to the same bus.

In order to allow messages to be transmitted on an exclusive access bus without conflict, a simple timing mechanism can be designed using the reflection code. That is, the step at which a node should transmit its message is determined by its identification (p, q, l). Here we use type 3 routing, which corresponds to the worst-case overhead, to illustrate how this can be achieved. The cases for types 2 and 3 are similar. Observe from Figs. 7, 8, and 9 that nodes of at most two consecutive levels in the X-tree are mapped to the same bus in the target array. Thus, the access of a bus has to be arbitrated only among these nodes. Specifically, in Type 3 routing, to transmit a message to a neighbor in the same row (the case for columns is similar), node (p, q, l), where l is even, transmits its message at the $(q+1)$th step, and node $(p', q', l-1)$ transmits its message at the $(p'+1+C^{l/2})$th step. Note that $C^{l/2} \leq C^{(L-1)/2}$ and $p' + 1 \leq C^{(L-3)/2}$. Thus, $(p' + 1 + C^{l/2}) \leq C^{(L-1)/2} + C^{(L-3)/2} = N_M$. (The reader may check row 1 (second row) in Fig. 8(b) for an example.) Using this bus arbitration technique, one type 3 routing step in the X-tree can be emulated in at most $C^{(L-1)/2} + C^{(L-3)/2}$ steps in the target array with exclusive access buses.

The above analysis shows that X-trees can be emulated much more efficiently on optical buses than on exclusive access buses. Note that the emulation of each of the three types of X-tree routing requires message transfers on both row and column buses in the target array. If we assume that message transmissions on row and column buses must be done in separate steps in the target array, then the number of emulation steps computed above doubles.

In the following, we analyze the alignment and expansion costs for M_1. Because embedding M_1 aligns all neighboring nodes of X-trees, its alignment cost, denoted $\Phi_{a,1}$, is optimal. That is,

$$\Phi_{a,1} = 0$$

To compute the expansion cost, $\Phi_{e,1}$, of embedding M_1, note that $\Phi_{e,1}$ can be improved without affecting the alignment cost by flipping over levels $L-4$ through 0 as shown in Fig. 7(b). It can be shown that in the case of optical buses, such flipping does not affect the efficiency for emulating X-trees on target arrays. In the case of exclusive access buses, however, the efficiency will be reduced, because in this case, $N_M = C^{(L-1)/2} + C^{(L-3)/2} + C^{(L-5)/2}$.

Assume that the embedding of an L-level X-tree (L odd) occupies m rows and n columns in Fig. 7(b). Then $m = C^{(L-1)/2} + C^{(L-3)/2} + C^{(L-5)/2}$ and $n = C^{(L-1)/2} + C^{(L-3)/2}$. Thus,

$$\Phi_{e,1} = \frac{m \times n}{|V_S|} = \frac{C^4 + C^3 - C - 1}{C^4 - C^{4-L}},$$

from which it is easy to show that $\Phi_{e,1}$ is asymptotically equal to 1.31 for $C = 2$ and 1.23 for $C = 4$. It is noted that with proper node labeling, the optimal alignment cost can also be achieved by using the well-known H-embedding in the case of X-binary-trees. The corresponding expansion cost, however, will be asymptotically equal to 2, which is much higher than that of M_1.

The above results indicate that embedding M_1 achieves an optimal alignment cost, but its expansion cost is not optimal. In the next section, we present our second embedding of X-trees, which achieves an optimal expansion cost. That embedding, however, does not align all neighboring nodes in X-trees.

IV. An X-Tree Embedding With Optimal Expansion Cost

In this section, our second embedding of L-level (L may be either odd or even) X-trees into 2-D processor arrays with spanning buses is presented. For this embedding, we assume that the number of nodes in each row of the target array is $n = C^\pi$, where π is an integer parameter in the range $1 \leq \pi < L$. Like the first embedding M_1 presented in the previous section, our second embedding is again obtained by mapping each level of an X-tree into a target subarray of proper size so that the alignment condition H1 is satisfied. The resulting target subarrays are then properly positioned to form the target array (see Fig. 10) so that H2 is satisfied, with the exception for the parent nodes on level $\pi - 1$ and their children. In this embedding, levels 0 through $\pi - 1$ of the X-tree are mapped to row 0 of the target 2-D array; level π to row 1 of the target array; level $\pi + 1$ to rows 2 through $C + 1$; and, in general, level l, $l \geq \pi$, of the X-tree to $C^{l-\pi}$ consecutive rows of the target array. Thus, the number of rows in the target array is $m = 1 + \sum_{l=\pi}^{L-1} C^{l-\pi} = (C^{L-\pi} + C - 2)/(C - 1)$. Note that when $n = 1$ or $m = 1$, corresponding to $\pi = 0$ and $\pi = L$, respectively, the target array becomes a linear array, and the problem of aligning neighboring nodes is trivial.

Let (r, l) denote the node at level l, $0 \leq l < L$, in the L-level X-tree with reflection index r. The embedding is defined by a mapping $M_2(r, l) = (u, v)$, which maps each node (r, l) in the X-tree to a node at row/column position (u, v) in the target array, as given below:

$$M_2(r,l) = \begin{cases} \left(0, C^{\pi-l}r + \frac{1}{2}\left(C^{\pi-l} - C\right)\right), & 0 \leq l < \pi \\ \left(r \bmod C^{l-\pi} + \zeta_l, \lfloor \frac{r}{C^{l-\pi}} \rfloor\right), & \pi \leq l < L \end{cases}, \quad (10)$$

where $\zeta_l = 0$ if $0 \leq l < \pi$, and $\zeta_l = (C^{l-\pi} + C - 2)/(C - 1)$ if $\pi \leq l < L$. Figs. 3(b) and 4(b) show the embedding M_2 for the X-trees in Figs. 3(a) and 4(a), respectively.

Theorem 2: The X-tree embedding M_2 defined in (10) satisfies the alignment conditions H1 and H2, except that the parents on level $\pi - 1$ and their children are not aligned.

The formal proof of this theorem is rather involved and is given in [11]. The analysis for the overhead incurred in emulating X-trees using embedding M_2 is similar to the case of M_1. It is noted that the emulation will not be as efficient, because not all neighboring nodes in X-trees are aligned in M_2. But it can be shown that at most one message relay by each node in the target network is necessary to emulate the routing between any two misaligned nodes. Thus, the increase in emulation overhead compared with M_1 is not significant. Such analysis is omitted because of space limitation.

In the following, we analyze the expansion cost, denoted $\Phi_{e,2}$, for embedding M_2. Because $|V_T| = m \times n = C^\pi(C^{L-\pi} + C - 2)/(C - 1)$ and $|V_S| = (C^L - 1)/(C - 1)$, we have:

$$\Phi_{e,2} = \frac{|V_T|}{|V_S|} = \frac{C^\pi(C^{L-\pi} + C - 2)}{C^L - 1} = \frac{C^L + C^\pi(C - 2)}{C^L - 1}.$$

When $C = 2$, corresponding to the case of X-binary-trees, we obtain:

$$\Phi_{e,2} = \frac{C_L}{C^L - 1} = \frac{|V_S| + 1}{|V_S|}, \quad C = 2.$$

That is, only one node is wasted, regardless of the choice of π. When $C = 4$, corresponding to the case of X-quad-trees, the expansion cost depends on the choice of π. Because it is desirable to make the target array more square, in general, we would like to choose $\pi = \lfloor L/2 \rfloor$.

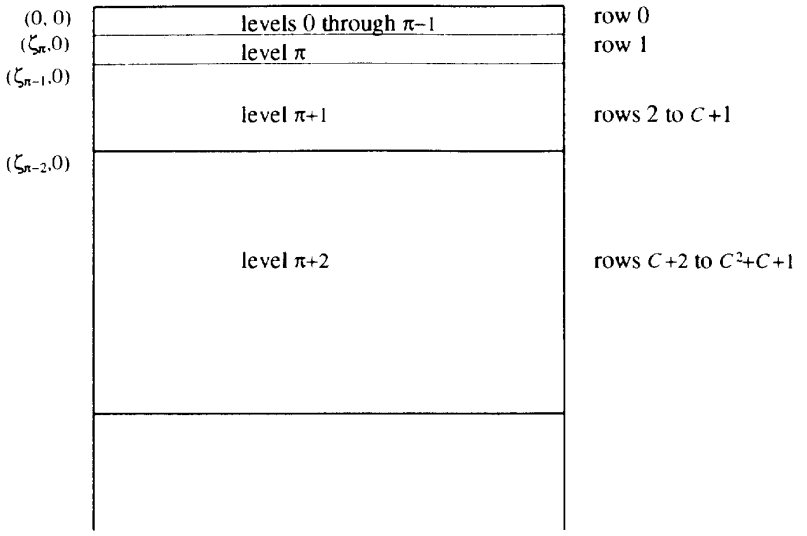

Fig. 10. Embedding M_2 of an L-level X-tree obtained by embedding individual levels of the X-tree into a square or rectangular area in a 2-D array with spanning buses.

Then:

$$\Phi_{e,2} = 1+\frac{1}{C^L-1}+\frac{C-2}{C^{\lceil L/2\rceil}-C^{-\lfloor L/2\rfloor}}, \quad C=4 \text{ and } \pi = \lfloor L/2\rfloor,$$

from which it is easy to show that $\Phi_{e,2} = 1.03$ for $L = 5$, $\Phi_{e,2} = 1.002$ for $L = 9$, and $\Phi_{e,2}$ approaches 1 as L goes large. It is noticed that when the target network is restricted to a square array, the unity expansion cost is not achievable by any embedding, because the number of nodes in an X-quad-tree is $(4^L-1)/3$, which is not a perfect square of any integer.

To compute the alignment cost of M_2, note from Theorem 2 that M_2 aligns all neighboring nodes in an X-tree, except the C^π nodes on level π and their parent. Thus:

$$\Phi_{a,2} = \frac{2(C-1)C^\pi}{C^L-1} = \frac{2(C-1)}{C^{\lceil L/2\rceil}-C^{-\lfloor L/2\rfloor}}, \quad \pi = \lfloor L/2\rfloor.$$

It is easily shown that if $L = 5$, then $\Phi_{a,2}$ is 0.26 for $C = 2$ and 0.09 for $C = 4$; and if $L = 9$, then $\Phi_{a,2}$ is 0.06 for $C = 2$ and 0.006 for $C = 4$. Recalling that Φ_a is in the range $[0, \delta_S]$, where $\delta_S = 2C+1$ is the degree of X-trees, these results are close to optimal.

V. Summary

Two embeddings of X-trees in 2-D processor arrays with spanning buses are presented in this short note. The first embedding has an optimal alignment cost, but its expansion cost can be as high as 1.31 and 1.23 for binary X-trees and pyramids, respectively. The second embedding asymptotically achieves an optimal expansion cost, but does not align all neighbors in the embedded X-tree. This demonstrates a trade-off between the expansion cost and the alignment cost for graph embeddings in processor arrays with spanning buses.

References

[1] S. H. Bokhari, "Finding maximum on an array processor with a global bus," *IEEE Trans. Comput.*, vol. C-32, no. 2, pp. 133–139, Feb. 1984.

[2] V. Cantoni and S. Levialdi, *Pyramidal Systems for Computer Vision, NATO ASI Series F: Computer and Systems Sciences.* New York: Springer-Verlag, 1986.

[3] Y. C. Chen, W. T. Chen, G. H. Chen, and J. P. Sheu, "Designing efficient parallel algorithms on mesh-connected computers with multiple broadcasting," *IEEE Trans. Parallel Distrib. Syst.*, vol. 1, no. 2, pp. 241–245, Feb. 1990.

[4] J. Cooper and S. G. Akl, "Efficient selection on a binary tree," *Inform. Processing Lett.*, vol. 23, no. 3, pp. 123–126, Mar. 1986.

[5] E. Dekel and S. Sahni, "Binary trees and parallel scheduling algorithms," *IEEE Trans. Comput.*, vol. C-32, no. 3, pp. 307–315, Mar. 1983.

[6] A. M. Despain and D. A. Patterson, "X-Tree: A tree structured multiprocessor computer architecture," in *Proc. 5th Int. Symp. Comput. Arch.*, 1978, pp. 144–151.

[7] P. W. Dowd, "High performance interprocessor communication through optical wavelength division multiple access channels," in *18th Int. Symp. Comput. Arch.*, 1991, pp. 96–105.

[8] Z. Guo and R. G. Melhem, "Embedding pyramids in array processors with pipelined buses," in *Proc. Int. Conf. Application Specific Array Processors*, 1990, pp. 665–676.

[9] Z. Guo, R. G. Melhem, R. W. Hall, D. M. Chiarulli, and S. P. Levitan, "Pipelined communications in optically interconnected arrays," *J. Parallel Distrib. Computing*, vol. 12, pp. 269–282, 1991.

[10] Z. Guo, "Sorting on array processors with pipelined buses," in *Proc. Int. Conf. Parallel Processing*, vol. III, pp. 289–292, 1992.

[11] Z. Guo and R. G. Melhem, "Embedding binary X-trees and pyramids in processor arrays with spanning buses," Tech. Rep. TR-EE-9208a, Dept. of Elec. Eng., Louisiana Tech Univ., 1992.

[12] J. M. Jolion and A. Rosenfeld, "An $O(\log n)$ pyramid Hough transform," *Patt. Recognition Lett.*, vol. 9, pp. 343–349, May 1989.

[13] M. Karnaugh, "A map method for synthesis of combinational logic circuits," *Trans. AIEE, Commun. and Electron.*, vol. 72, no. 1, pp. 593–599, Nov. 1953.

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

[15] R. G. Melhem, D. M. Chiarulli, and S. P. Levitan, "Space multiplexing of waveguides in optically interconnected multiprocessor systems," *Comput. J.*, vol. 32, pp. 362–369, Apr. 1989.

[16] V. K. Prasanna-Kumar and C. S. Raghavendra, "Array processor with multiple broadcasting," *J. Parallel Distrib. Computing*, vol. 4, pp. 173–190, 1987.

[17] V. K. Prasanna-Kumar and D. Reisis, "Image computations on meshes with multiple broadcast," *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. 11, pp. 1194–1202, Nov. 1989.

[18] C. Qiao and R. G. Melhem, "Reconfiguration with time division multiplexed MIN's for multiprocessor communications," *IEEE Trans. Parallel Distrib. Syst.*, vol. 5, pp. 337–352, Apr. 1994.

[19] Q. F. Stout, "Mesh connected computers with broadcasting," *IEEE Trans. Comput.*, vol. C-32, no. 9, pp. 826–830, Sept. 1983.

[20] C. D. Thompson and H. T. Kung, "Sorting on a mesh-connected parallel computer," *Commun. ACM*, vol. 20, no. 4, pp. 263–271, Apr. 1977.

[21] T. S. Wailes and D. G. Meyer, "Multiple channel architecture: A new optical interconnection strategy for massively parallel computers," *J. Lightwave Technol.*, vol. 9, pp. 1702–1716, Dec. 1991.