Efficient Embeddings of Binary Trees in VLSI Arrays

DAN GORDON

Abstract—**We consider the problem of embedding a complete binary tree in square- or hexagonally-connected VLSI arrays of processing elements (PE's). This problem can be solved in a radically different manner from current layout techniques which are aimed at laying out a given graph in the plane. The difference is due to the fact that a PE can be used both as a tree node and as a connecting element between distant nodes. New embedding schemes are presented in which (asymptotically) 100 percent of the PE's are utilized as tree nodes. This is a significant savings over known schemes, which achieve 50 percent utilization (the well-known H-tree) and 71 percent for some hexagonal schemes. These schemes also speed up signal propagation from the root to the leaves.**

Index Terms—**Area-efficient embedding, arrays of processing elements, binary tree layout, graph embedding, interconnection network, VLSI.**

I. Introduction

VLSI technology [15] enables us to construct large pieces of silicon containing many processing elements (PE's), interconnected in various ways. Many patterns of interconnection have been proposed for solving a variety of problems [1], [13], [15], [20], [26], [27]. Some typical patterns are linear arrays, rectangular and hexagonal arrays, binary trees, X-trees, etc. The problem of laying out a given pattern in the plane has been studied extensively, both for particular graphs and for general classes of graphs, such as planar graphs [1]-[6], [8], [9], [11]-[18], [20], [21], [23]-[28].

Once a certain pattern is hardwired into the silicon, it serves its particular purpose well, but it is less useful for other applications. One is thus led to consider some fixed hardwired pattern of PE's onto which other patterns can be embedded by dynamic reconfiguration of the inter-PE connections. This approach, proposed in [9], increases the cost effectiveness of a given piece of silicon by increasing its range of applications. Two obvious choices for the fixed pattern are the square and hexagonal grids due to their own usefulness, regularity, and locality of interconnection. Such patterns are commonly called *arrays* of PE's.

The design rules that have been followed for laying out a graph in the plane were to map nodes of the graph onto regular grid points and edges onto the line segments connecting the corresponding points [2], [4], [5], [8], [11], [13]-[16], [18], [20], [21], [23], [24], [27], [28]. Every such layout trivially becomes an embedding for the graph in an *array* of PE's: nodes are mapped onto PE's and edges onto one or more inter-PE connections. The problem with this approach is that when an edge goes through several PE's, these act only as connectors and their computational ability is lost. A typical example is the H-tree embedding of a complete binary tree [1], [8], [13], [16], [24]. (See Fig. 1.) Our primary motivation stems from the need to minimize this waste, which is especially significant when the granularity (processing power) of a PE is large.

In this paper we demonstrate that a radically different approach can be used to tackle the embedding problem in an array of PE's. In our approach, a PE can be used both as a node in the graph and as a connecting element between distant nodes. We study the particular (but important) problem of the complete binary tree and show that it is possible to utilize (asymptotically) 100 percent of the PE's as tree nodes. This is in marked contrast to the 50 percent utilization of the H-tree and 71 percent utilization of some hexagonal patterns [6]. Our concepts are potentially useful also for all graphs for which optimal embeddings are already known in the big-O sense.

The rest of the paper is organized as follows. Section II briefly summarizes some previous work on graph layouts and embeddings. Section III shows why connectors are necessary and presents the theoretical framework for our schemes. In Section IV, we present our scheme for hexagonal arrays, in which all PE's (except one) are used as tree nodes. Section V deals with square-connected arrays, for which 100 percent utilization is much more difficult, both to design and to prove correct. In Section VI we give an area and propagation-delay analysis of our schemes, and compare them to previous schemes. We also derive geometric lower bounds on the delay. Section VII concludes with some further research possibilities.

II. Previous Work

One important area in which dynamic reconfiguration has been used, and which we do not address in this paper, is the issue of fault-tolerance—see, for example, [7], [9], [10], [19], [22].

A different approach from ours to dynamic reconfiguration is the CHiP computer [25], in which reconfigurable switches are positioned between the PE's, and extra bus lines run horizontally and vertically between the PE's. In our approach, the switching is done by the PE's and there are no extra bus lines.

The problem of laying out a graph in the plane (as opposed

Manuscript received April 8, 1985; revised March 18, 1987.
The author was on leave at the Department of Computer Science, University of Cincinnati, Cincinnati, OH. He is now with the Department of Mathematics and Computer Science, University of Haifa, Haifa 31999, Israel.
IEEE Log Number 8715297.

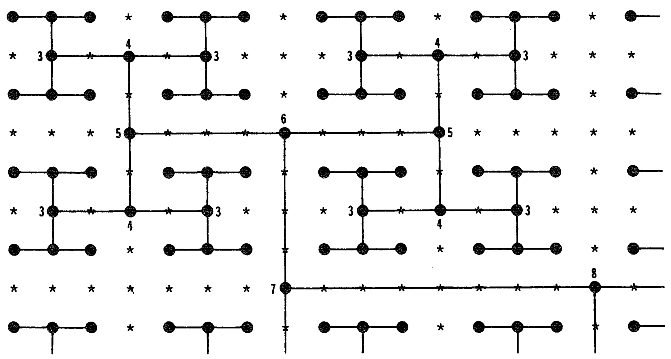


Fig. 1. The H-tree layout for square arrays (PE's serving as nodes in the tree are enlarged).

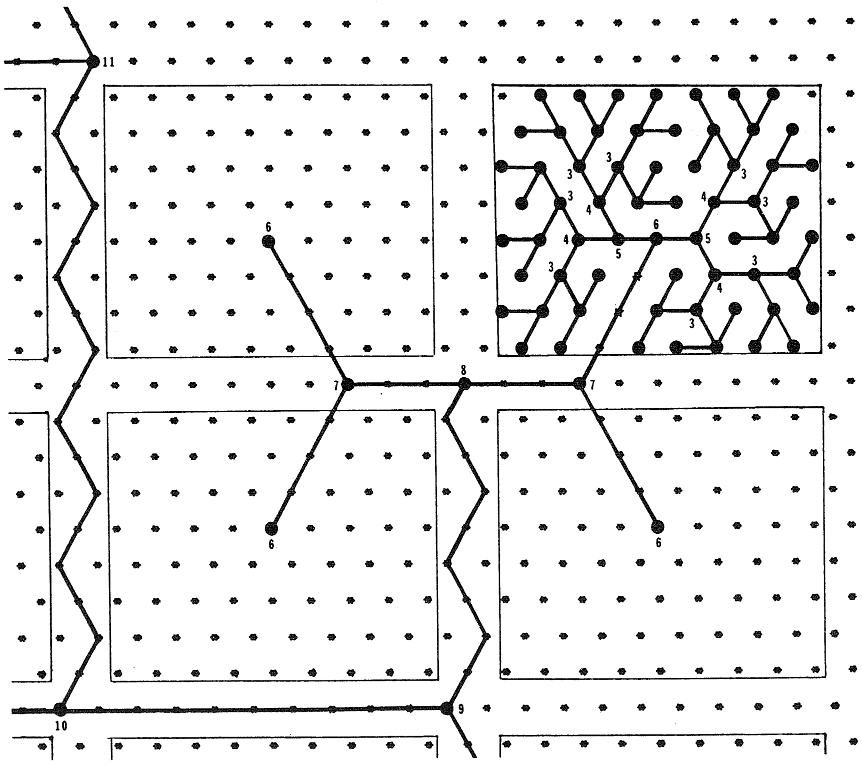


Fig. 2. The rectangular embedding scheme of [6] for hexagonal arrays.

to embedding it in an array of PE's) has been studied extensively by many researchers—see, for example, [1]–[6], [8], [9], [11], [12]–[18], [20], [21], [23], [24], [27], [28]. The following are some of the important special issues that were studied.

Minimizing the Longest Edge Length: This problem was shown to be NP-complete for general graphs [17], and a class of graphs was found which exhibit a tradeoff between the layout area and the maximal edge length [3]. Bounds for the maximum edge length of complete binary trees can be found in [21], while [23] uses the "hyper-H" scheme to obtain the minimal (and optimal) edge length of $O(\sqrt{n}/\log n)$ for arbitrary binary trees. See [5], [15], [21], [27] for discussions on the importance of minimizing edge length.

Access to All Leaves: For some applications, it is necessary to access all leaves of the tree directly from the boundary. When leaves are restricted to lie on the boundary of a convex region then $\Omega(n \log n)$ area is necessary (and sufficient) to lay out a complete binary tree of n leaves [4]. Other schemes for layouts with access to the leaves can be found in [11] and [18]. We note here that if all leaves can be accessed and the layout shape can be enclosed in a rectangle of *bounded aspect ratio,* then $\Omega(n^2)$ area is required.

Minimum Area Layouts: Arbitrary binary trees of n nodes can be laid out in $O(n)$ area [8], [28], and this is obviously optimal. The H-tree layout for complete binary trees appears in many references, the earliest being apparently [1], [13], [16], [24]. Various schemes for hexagonal arrays can be found in [6].

III. Theoretical Development

In this section we present the theoretical motivation and framework for our embedding schemes. Fig. 1 shows the well-known H-tree embedding for square arrays and Fig. 2 shows one of the patterns of [6] for hexagonal arrays. In these schemes, some PE's are not even used as connectors. The first question to be raised is why connectors are at all necessary, and if so, how many. This is answered in the following

Theorem 1: Assume that an interconnection network of PE's is laid out in the plane so that PE areas are bounded from below by a constant, and the distances between (the centers of) connected PE's are bounded from above by a constant. Then any embedding of a complete binary tree of n nodes requires $\Omega(n)$ connectors.

Proof: In a circle of (Euclidean) radius r from the tree's root, the number of PE's is $O(r^2)$, but for a complete binary tree without connectors, the number of PE's must be $\Omega(2^r)$. Therefore, there exists a constant c which is the maximal level of tree that can be embedded in the given plane of PE's without connectors. In a tree of n nodes, with $n = 2^m - 1$ and $m \geq c + 1$, the number of connectors is at least the number of subtrees of level $c + 1$, which is $2^{m-c-1} = 2^{-c-1}(n + 1) = \Omega(n)$. □

Theorem 1 applies to all regular grid patterns, as well as to irregular ones satisfying the conditions of Theorem 1. It shows that connectors take up a fixed percentage of the tree nodes in any interconnection network, so we consider PE's capable of being nodes and connectors at the same time.

Our approach is based on embedding a complete binary tree of some fixed level c in a "tile" containing $t \geq 2^c$ PE's. This leaves each tile with at least one extra PE to be used as a node on some higher level of the tree. The idea of using an extra PE as a higher level node was used in [2], [14] to assemble trees of arbitrary levels out of one kind of chip. The tiles are to be placed adjacent to each other so that no PE's are wasted. We call the ratio $2^c/t$ the (asymptotic) *utilization ratio* of the scheme. All our embedding schemes are planar, even inside the PE (i.e., a PE does not cross connect neighbors and connections do not cross a father–son link).

IV. Hexagonal Arrays

For hexagonal arrays, we have three types of basic tiles, which we call A, B, C. Each basic tile contains 64 PE's, 63 for a six-level tree and one extra PE. The precise layout of the three basic tiles is shown in Fig. 3. In all the figures, communications passing through a PE (i.e., using the PE as a connector) are shown as being routed around the PE.

In the following description of the three types, we assume absolute directions in the plane of PE's, which we call up, down, left, and right (corresponding to the same directions in Fig. 3). We describe the tile types as if the path to the root of the tree goes upwards, although in the actual layout a tile may be rotated by 180° and the path would go downwards.

Type A: In this tile, the extra PE is located (approximately) at the center of the lower edge of the tile. It is always the parent of the tile's root.

Type B: The position of the extra PE is at the top right corner of the tile.

Type C: The extra PE is located at the bottom left corner of the tile.

For each given type, a tile of that type containing a $2m$-level tree can be constructed from four tiles of level $2(m - 1)$, for

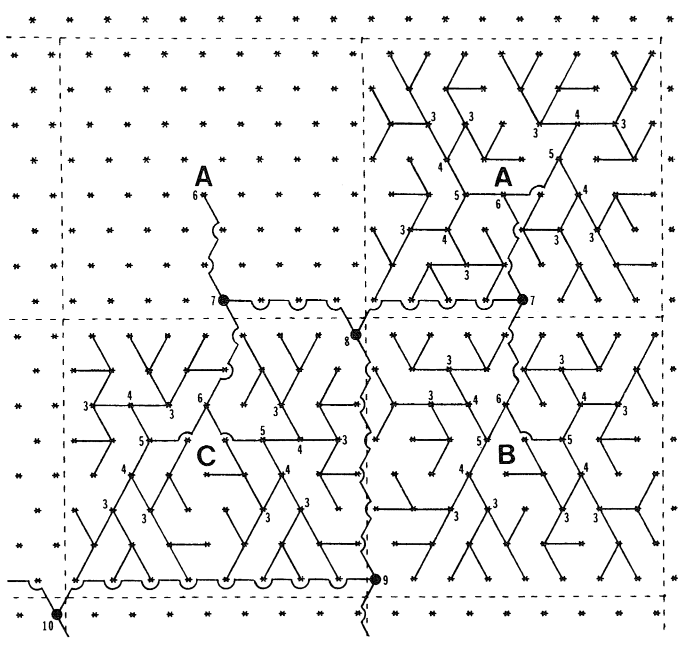


Fig. 3. The three basic tiles—A, B, C—for hexagonal arrays. Shown in the figure is an A tile of level 8 made up of four basic tiles, with connections to higher level nodes.

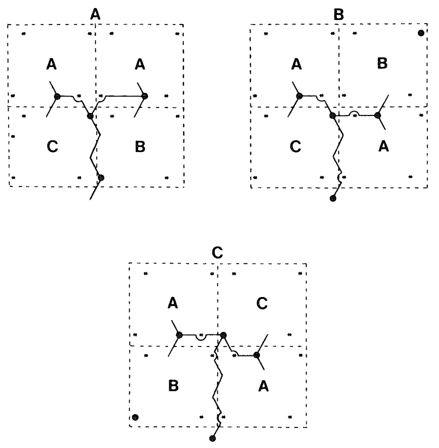


Fig. 4. Schematic description of the recursive construction of the three tile types for hexagonal arrays.

$m \geq 4$. Fig. 4 is a schematic description of the recursive construction of the three types. A $(2m - 1)$-level tree can clearly also be constructed out of two $(2m - 2)$ tiles.

An A tile is made up of two smaller A's on top, a C on the lower left, and a B on the lower right. C's extra PE is the root of the larger A, and B's extra PE is the large A's extra PE (which is the root's father node). Note that Fig. 3 is an eight-level tile of type A made up of four basic tiles. A B tile is made up of two A's, a B, and a C as shown. C's extra PE is the large B tile's root, and the small B's extra PE is also the larger B's extra PE. C is also made up of two A's, a B, and a C as shown. The small C's extra PE is the root, and the small B's extra PE is the larger C's extra PE.

Two points should be noted about the long communication lines. The vertical communications are made up as shown in Fig. 3 between the B and C tiles. This type of path is possible between any two basic tiles, including A, so it is also possible between tiles of a higher level. The second point concerns horizontal communications. In the three schemes of Fig. 4, there is one A tile to the left and one A tile to the right of the root. The horizontal communications from the root to its sons always pass through the bottom row of the A tile (as in Fig. 3). This means that the tree of an A tile of any level may not use any links on the bottom row of the tile. From Fig. 3, we see that no horizontal links are used by a tree in any top or bottom row of a basic tile. Therefore, the horizontal links on the bottom row of an A tile (of any level) always remain free to accommodate high-level communications.

V. Square-Connected Arrays

A. Preliminaries

The problem here is harder than the hexagonal case because only the leaves can serve as connectors. The approach that we have found useful is for the boundaries between tiles to pass through rows and columns of PE's, with the PE's on the boundaries assigned to one of the two bordering tiles.

An $i \times j$ *scheme* is one in which an $(i + 1)(j + 1)$ rectangle of PE's contains a basic tile. Only half the PE's on the boundaries and only one corner PE should be counted as belonging to the tile. So the number of PE's assigned to the tile is $[(i - 1)(j - 1)$ interior PE's$] + [(i - 1) + (j - 1)$ border PE's$] + [1$ corner PE$] = ij$ PE's. If a basic tile contains a tree of level c, then the utilization ratio is $2^c/ij$. Note that about half the PE's on the outer boundary, $\Theta(\sqrt{n})$, are left unused (n is the total number of nodes).

B. Two Simple Schemes

Due to the complexity of our 100 percent scheme, we introduce first two simple schemes with around 90 percent utilization. The first one is the 3×3 scheme shown in Fig. 5. A basic tile contains a tree of level 3, and thus the utilization ratio is $2^3/9 = 88.89$ percent. Note that all the basic tiles are the same, the only difference being in the path from a tile's root to its parent node.

The second scheme is the 5×7 one shown in Fig. 6. A basic tile contains a five-level tree, yielding a utilization ratio of $2^5/35 = 91.43$ percent, which is slightly better than the 3×3 scheme. Again, the basic tile is always the same, with only a different path from a tile's root to its parent node.

C. 100 Percent Utilization—Informal Outline

In order to achieve 100 percent utilization, all PE's must be used, including the corners of the tiles. Some of the corner PE's are naturally used as nodes on a higher level (i.e., a level higher than the root of a basic tile). For convenience, we refer to such higher level nodes as *high-nodes*. The problem with a corner PE which is not a high-node is determining which of the four bordering tiles it should belong to.

Fig. 7 is a schematic embedding in which each square represents one basic tile and the high-nodes are enlarged. Our solution to the above problem is based on the observation that the high-nodes are distributed more densely towards the lower levels of the tree (because the paths between them get shorter as the level decreases). This fact suggests that in assigning corner PE's to tiles, priority should be given to tiles in the direction of the *lower* high-nodes, because these tiles have less free corners to draw on. This is done as follows.

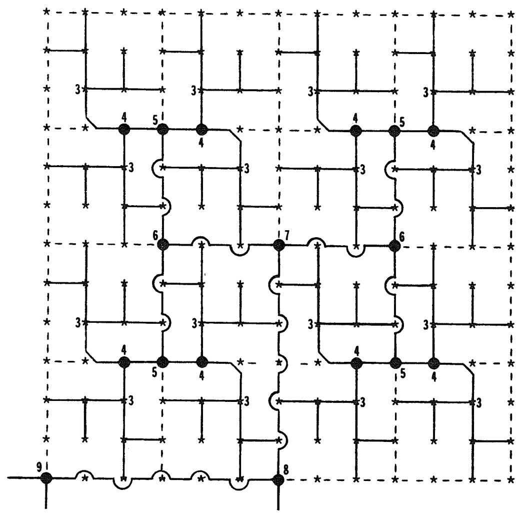


Fig. 5. The 3 × 3 scheme for square arrays achieving 88.89 percent utilization.

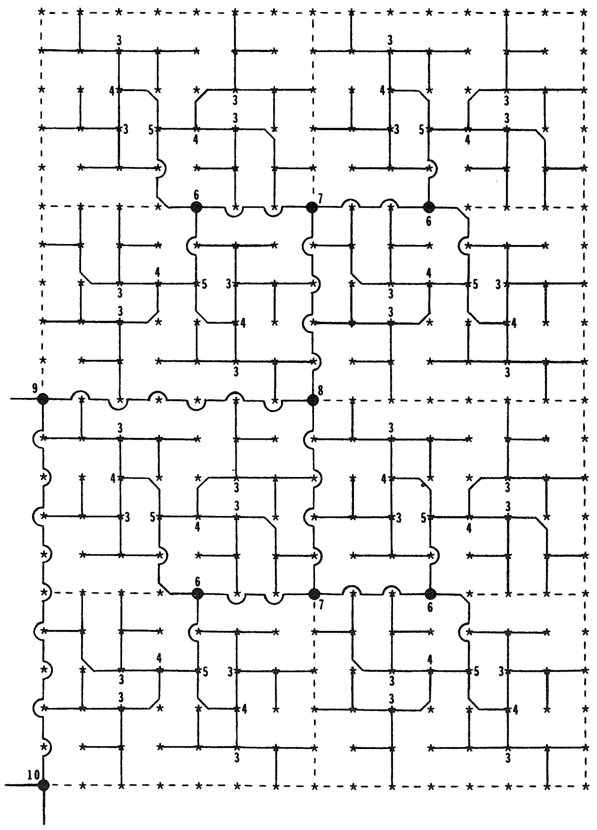


Fig. 6. The 5 × 7 scheme for square arrays achieving 91.43 percent utilization.

Each corner PE which is not a high-node is either on a communicating path between two high-nodes, or on the reflection of such a path—the reflection taken about the lower of the two high-nodes. The corner PE is assigned to a tile in the direction of the lower high-node, and so is its reflected PE. The reflected PE, though, is assigned to a tile on the other side of the (reflected) path. In Fig. 7, the short diagonal line at the

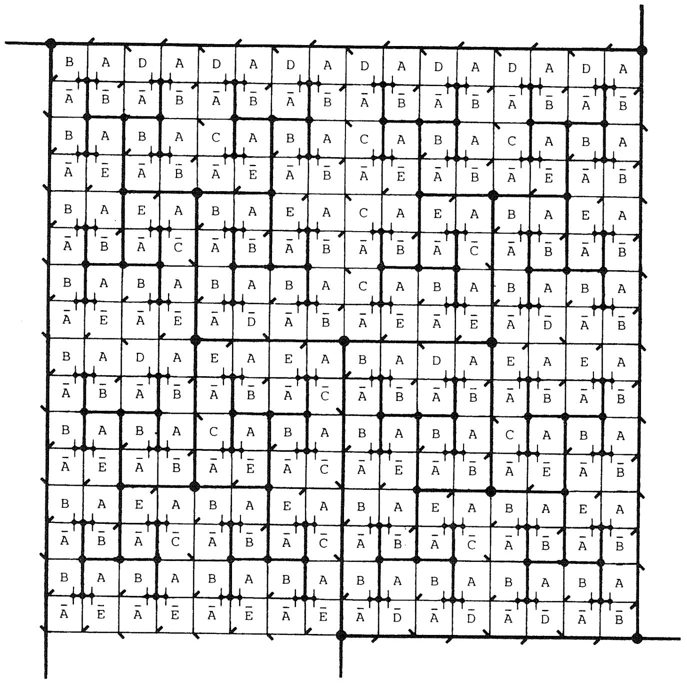

B A D A D A D A D A D A D A D A
Ā B̄ Ā B̄ Ā B̄ Ā B̄ Ā B̄ Ā B̄ Ā B̄ Ā B̄
B A B A C A B A C A B A C A B A
Ā Ē Ā B̄ Ā Ē Ā B̄ Ā Ē Ā B̄ Ā Ē Ā B̄
B A E A B A E A C A E A B A E A
Ā B̄ Ā C̄ Ā B̄ Ā B̄ Ā B̄ Ā C̄ Ā B̄ Ā B̄
B A B A B A B A C A B A B A B A
Ā Ē Ā Ē Ā D̄ Ā B̄ Ā Ē Ā Ē Ā D̄ Ā B̄
B A D A E A E A B A D A E A E A
Ā B̄ Ā B̄ Ā B̄ Ā C̄ Ā B̄ Ā B̄ Ā B̄ Ā B̄
B A B A C A B A B A B A C A B A
Ā Ē Ā B̄ Ā Ē Ā C̄ Ā Ē Ā B̄ Ā Ē Ā B̄
B A E A B A E A B A E A B A E A
Ā B̄ Ā C̄ Ā B̄ Ā C̄ Ā B̄ Ā C̄ Ā B̄ Ā B̄
B A B A B A B A B A B A B A B A
Ā Ē Ā Ē Ā Ē Ā Ē Ā D̄ Ā D̄ Ā D̄ Ā B̄

Fig. 7. Schematic layout of the 8 × 8 scheme showing positions of the basic tiles (A-E). The corner PE's are assigned to the tiles with the short diagonal lines.

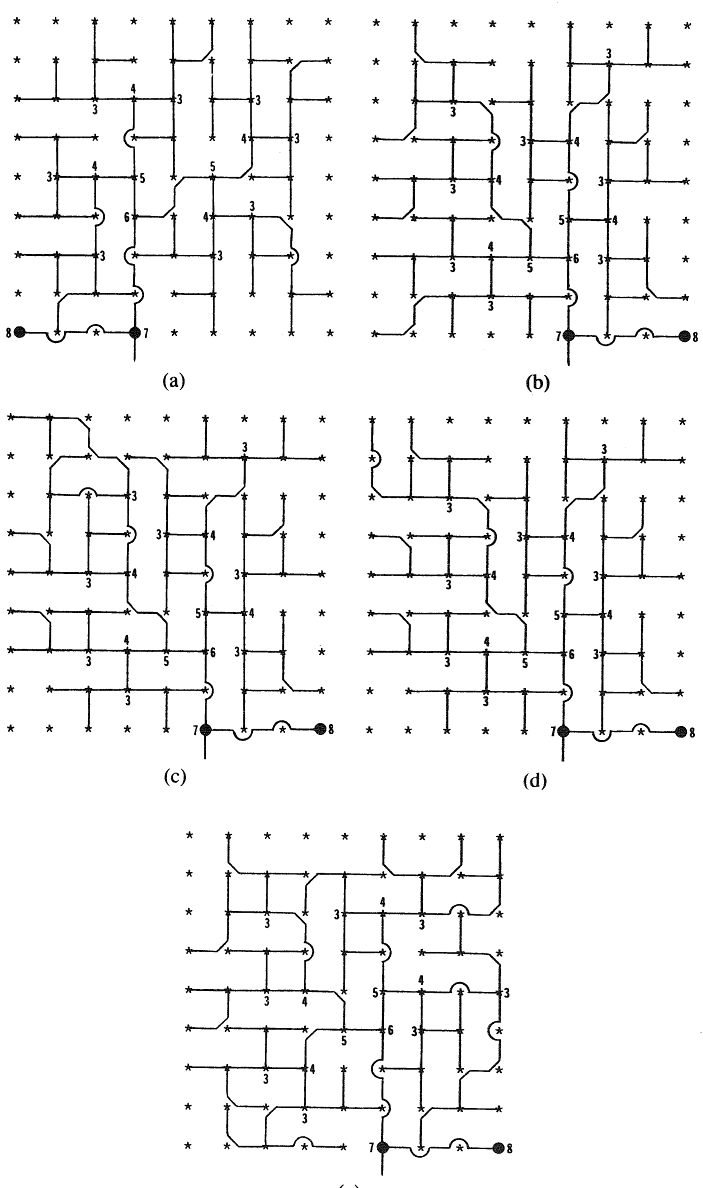


Fig. 8. The five interlocking basic tiles (A-E) of the 8 × 8 scheme, achieving 100 percent utilization. (a) Type A—no corner PE's. (b) Type B—lower left PE with horizontal access. (c) Type C—top left PE with horizontal access. (d) Type D—top left PE with vertical access. (e) Type E—top right PE with vertical access.

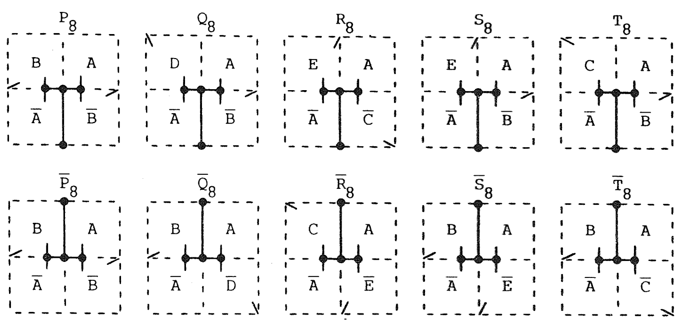


Fig. 9. Construction of tiles P_8-T_8 out of A-E showing assignments of the corner PE's.

corner of a tile indicates that the corner PE is assigned to that tile.

Once a corner PE is assigned to a tile, it remains to determine the direction of access from the tile to the corner. If the corner PE is on a communicating path between high-nodes, it can only be accessed in a direction perpendicular to that path. Corner PE's which are on a reflected path could be accessed from their tile in two directions, but it is simpler to consider the reflected path as real and allow access only in the perpendicular direction.

All the above leads to five different basic tiles, shown in Fig. 7 as A, B, C, D, E. The rotated tiles are marked $\bar{A}$-$\bar{E}$. Each tile is characterized by the corner PE it gets (if any) and by the direction of access to the corner PE. Type A is accessed from the left, all others from the right. Type A has no corner PE's; B has the lower left PE with horizontal access; C and D have the upper left PE with horizontal and vertical access, respectively; E has the upper right PE with vertical access. It is clear from the figure that the tiles should be designed so that any of B-E can be placed on either side of A and below or above $\bar{A}$.

The above characterizations of A-E can be realized by tiles of size 8 × 8, shown in Fig. 8. Each tile contains a six-level tree, so the utilization ratio is $2^6/64 = 100$ percent.

D. 100 Percent Utilization—Recursive Construction and Correctness

The five basic tiles A-E do not extend recursively to higher levels as in the hexagonal case. To this end, we need to construct five new tiles, P_8-T_8 out of A-E as shown in Fig. 9. The short diagonal lines at the corners show which of the smaller tiles they belong to; the direction of access to a corner is through the path making the *smaller* angle with the diagonal line. The rotated tiles, $\bar{P}_8$-$\bar{T}_8$, are also shown.

P_8-T_8 can be used to build up tiles of similar types of any desired level. The recursive construction is shown in Fig. 11, where the diagonal lines have the same significance as above. However, this construction and its correctness require the introduction of a special symbolism to handle statements about tiles and binary trees.

We formally define *a tile X of level m,* where $m \geq 8$ and even, as a square array of PE's with $2^{m/2} + 1$ PE's to a side, X has embedded in it a complete binary tree of level m, with the tree's root connected to BC(X) = the PE at the center of X's bottom row. We refer to X's bottom left, top left, top center, top right, and bottom right PE's in a similar abbreviated manner. X's four sides are called TOP(X), BOT(X), LEFT(X) and RIGHT(X). NODES(X) denotes the set of all PE's which are used as nodes in X's graph (the tree augmented by BC(X)). LINKS(X) is the set of all inter-PE links which are used as edges or parts of edges in X's graph. $\bar{X}$ denotes X rotated by 180°. P_8-T_8 are examples of tiles of level 8 (A-E are not tiles according to this definition and we reserve the term *basic tile* for them).

Some further notations: if a, b are two PE's on the same row or column, then $[a, b]$ denotes the set of all PE's between a and b, including a and b. A set L of PE's is called *free in X* if no links between any adjacent PE's of L belong to LINKS(X).

Definition: Let X and Y be tiles of the same level. We define three types of adjacency conditions:

$X|Y$ means that X can be placed adjacent to Y so that RIGHT(X) = LEFT(Y); all PE's on the common boundary, with the possible exception of the extremes, are nodes in the graphs of X or Y; and no node or link is in both X and Y. $\bar{X}|\bar{Y}$ is defined similarly.

$X/\bar{Y}$ is defined similarly, but with X placed over $\bar{Y}$ and BC(X) coinciding with TC($\bar{Y}$).

$\bar{X}/Y$ is defined in a similar manner (no common nodes, though).

Note that $X|Y \Leftrightarrow \bar{Y}|\bar{X}$.

Lemma 1: If X, Y, Z, U, are tiles of level m satisfying conditions 1)–9) below, then a tile of level $m + 2$ can be constructed out of X, Y, Z, U.

1) $X|Y$, $\bar{Z}|\bar{U}$, $Y/\bar{U}$, $X/\bar{Z}$; 2) BR($\bar{Z}$), TR($\bar{Z}$) $\notin$ NODES($\bar{Z}$); 3) BL($\bar{U}$), TL($\bar{U}$) $\notin$ NODES($\bar{U}$); 4) BR($\bar{X}$) $\notin$ NODES(X); 5) BL(Y) $\notin$ NODES(Y); 6) RIGHT($\bar{Z}$) and [TR($\bar{Z}$), TC($\bar{Z}$)] are free in $\bar{Z}$; 7) LEFT($\bar{U}$) and [TL($\bar{U}$), TC($\bar{U}$)] are free in $\bar{U}$; 8) [BR(X), BC(X)] is free in X; 9) [BL(Y), BC(Y)] is free in Y.

The proof is illustrated in Fig. 10. The tile constructed out of X, Y, Z, U is denoted by TILE(X, Y, Z, U). Our main result is

Theorem 2: For every $m \geq 8$ and even, a tile of level m can be constructed.

Proof: The following stronger assertion is proved by induction on m:

Assertion: For every $m \geq 8$ and even, we can construct tiles P, Q, R, S, T of level m so that

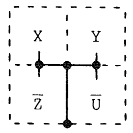


Fig. 10. Construction of a tile out of four lower level tiles X, Y, Z, U.

1) TL(Q) $\in$ NODES(Q), BR(R) $\in$ NODES(R), TL(T) $\in$ NODES(T); all other corner PE's are *not* nodes in the graphs of their respective tiles;

2) with the exception of LEFT(Q), BOT(R), TOP(T), all boundaries are free in their respective tiles; [BL(R), BC(R)] is free in R;

3) $P|Q$, $P|S$, $Q|Q$, $P|R$, $S|R$, $T|R$, $S|S$, $T|Q$, $R|P$, $Q|S$, $S|T$;

4) $P/\bar{S}$, $Q/\bar{P}$, $Q/\bar{R}$, $S/\bar{R}$, $R/\bar{T}$, $S/\bar{P}$, $T/\bar{R}$;

5) $\bar{S}/P$, $\bar{P}/R$, $\bar{R}/P$, $\bar{P}/S$, $\bar{R}/T$, $\bar{T}/R$, $\bar{S}/Q$, $\bar{Q}/S$.

P_8-T_8 form the induction basis, as can be verified by inspection. The induction hypothesis is that tiles P-T of level m can be constructed so that 1)–5) hold. We now construct the following tiles of level $m + 2$ as in Fig. 11: P' = TILE(P, Q, S, P), Q' = TILE(Q, Q, R, P), R' = TILE(S, R, R, T), S' = TILE(S, S, R, P), and T' = TILE(T, Q, R, P).

To verify that the above constructions are possible, note that from 1)–4) of the induction hypothesis it follows that P, Q, S, P satisfy the conditions for X, Y, Z, U, respectively in Lemma 1; hence, P' can be constructed. Similarly for Q'-T'.

The next stage is to show that conditions 1′)–5′) hold, where i′) is the same as i) but with P'-T' in place of P-T. Note first how the adjacencies in 3)–5) follow from the recursive construction: Since P' is TILE(P, Q, S, P), we must have $P|Q$, $\bar{S}|\bar{P}$ (or $P|S$), $P/\bar{S}$ and $Q/\bar{P}$. However, not all the adjacencies listed in 3)–5) are immediate, e.g., $R|P$ does not follow from any adjacency of R and P in one of P'-T', but since $P|Q$ must also hold for P' and Q', we see that $P'|Q'$ implies $Q|Q$ (which is already listed) and $\bar{P}|\bar{R}$ (or $R|P$). Also, the adjacencies of 5) are not apparent in the tiles P'-T', but follow from 4′): e.g., $P'/\bar{S}'$ implies $\bar{S}/P$ and $\bar{P}/R$. Actually, 4′) implies only the first six adjacencies in 5)—the other two follow from these six. It is possible to check that 3)–5) is a "complete" and minimal list of adjacencies in the following sense: any adjacency condition derived from the recursive construction of P-T, either directly or indirectly as above, is in 3)–5); and every adjacency condition in 3)–5) can be derived in this manner.

To see 1′), take note of 1) and observe that in Fig. 11 only TL(Q'), BR(R'), and TL(T') are nodes in their respective graphs. These three corners are, respectively, TL(Q), BR($\bar{T}$) (which is TL(T)), and TL(T), which are allowed exceptions in 1). To see 2′), note in Fig. 11 that all boundaries of P'-T', except for LEFT(Q'), BOT(R'), and TOP(T'), are made up of boundaries which are free in P-T according to 2). [BL(R'), BC(R')] is just BOT($\bar{R}$) or TOP(R), which is free in R.

For 3′), we show only $P'|Q'$—the other adjacencies can be done similarly. From 3) we have $Q|Q$ and $R|P \Rightarrow \bar{P}|\bar{R}$. Furthermore, the top, middle, and bottom PE's of RIGHT(P') do not belong to NODES(P') (since they are not in

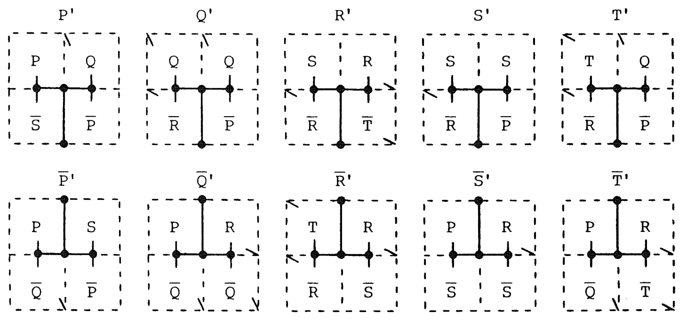


Fig. 11. Recursive construction of tiles P'–T' out of lower level tiles P–T, showing assignments of the corner PE's.

TABLE I
ASYMPTOTIC AREA UTILIZATION (PERCENTAGE OF PE'S USED AS NODES) OF THE NEW SCHEMES COMPARED TO EXISTING METHODS FOR SQUARE AND HEXAGONAL ARRAYS

Square	H-Tree	50%
	3x3 Scheme 5x7 Scheme 8x8 Scheme	88.89% 91.43% 100%
Hexagonal	Ref. [6]	71%
	New Scheme	100%

NODES(Q) or NODES($\bar{P}$)), so P' can be placed to the left of Q' without conflicting demands for any PE's on the common boundary. The middle PE of the common boundary belongs to NODES(Q'), and, hence, no interior PE is left unused.

The proofs of 4') and 5') are similar: $P'/\bar{S}'$ follows from $\bar{S}/P$ and $\bar{P}/R$ in 5) and from the fact that there are no conflicting demands for the left and right PE's of the common boundary; BC(P') and TC($\bar{S}'$) become the same node which belongs to both NODES(P') and NODES($\bar{S}'$). $\bar{S}'/P'$ follows from $\bar{S}/P$ and $\bar{S}/Q$ in 5), and from the fact that on the common boundary, there are no conflicts for the extreme PE's and the middle PE is in NODES(P') but not in NODES($\bar{S}'$). Similar arguments hold for the other adjacencies in 4') and 5'). □

VI. Area and Delay Analysis

A. Area Analysis

For a tree of n nodes, $n + 1$ PE's are used in the hexagonal scheme, so the precise utilization ratio is $n/(n + 1)$. The limit as $n \to \infty$ is 100 percent, compared to 71 percent for the hexagonal schemes of [6].

For square arrays, a tree of $n = 2^k - 1$ nodes requires a tile of level k with $\sqrt{n + 1} + 1$ PE's to a side. The precise utilization ratio is, therefore $n/(n + 2\sqrt{n + 1} + 2)$. Again, the limit as $n \to \infty$ is 100 percent compared to 50 percent for the H-tree [6]. For "practical" values of k, such as 10, 12, 14, 16, the utilization ratios are, respectively, 93.94, 96.92, 98.45, and 99.22 percent. Note that the hexagonal scheme is better for these values of k (over 99.90 percent). Table I summarizes the area utilization of the various methods.

The actual savings may be slightly smaller than indicated by the above figures because our "enhanced" PE (capable of being a node and a connector) may need more area than a "simple" PE (capable of only one function), due to the improved connection capability. The difference is probably small because even the simple PE's have to retain some logical capability when serving as connectors (e.g., when one wants to "undo" the tree). Note, however, that the extra area required for connections proportionally decreases as the granularity increases. Furthermore, in such a setting (high granularity), our savings become more significant because less area is wasted on connectors and idle PE's.

B. Delay Analysis

There are two ways in which a connector can work: one is by "closing a switch" (effectively linking wires) and another is as a signal enhancer. The first method is limited by the ability of a PE to drive a signal through a long wire, while the second introduces a longer delay. In our delay analysis, as in [6], we count the maximal number of links a signal has to cross in order to reach a leaf from the root of the tree. This approach is compatible with the second method, and any connectors acting as switches would shorten the delay. See [5], [15], [21] for issues of wire length and delay. Note that our model conforms to the one suggested in [5], in which delay is proportional to wire length due to the need for enhancers along long wires.

Let $D(k)$ denote the propagation delay for a tree of level k, with an appropriate subscript designating the scheme. The following recurrence relations for our hexagonal scheme are readily obtained: $D_{hx}(6) = 6$, $D_{hx}(7) = 7$, $D_{hx}(2k) = D_{hx}(2k - 1) + 2^{k-2} + 1$, and $D_{hx}(2k + 1) = D_{hx}(2k) + 2^{k-1}$.

TABLE II
PROPAGATION DELAY TIMES FOR TREES OF LEVEL m—COMPARISON OF THE NEW SCHEMES TO PREVIOUS METHODS FOR SQUARE AND HEXAGONAL ARRAYS

		m even	m odd
Square	H-Tree	$(1.5)2^{m/2}-2$	$(1.4142)2^{m/2}-2$
	3x3 Scheme	$(1.125)2^{m/2}-1$	$(1.0607)2^{m/2}-1$
	5x7 Scheme	$(1.0625)2^{m/2}-1$	$(1.0607)2^{m/2}-1$
	8x8 Scheme	$2^{m/2}$	$(1.0607)2^{m/2}$
Hexagonal	Ref. [6]	$(1.1875)2^{m/2}-4$	$(1.2374)2^{m/2}-4$
	New Scheme	$2^{m/2}+m/2-6$	$(1.0607)2^{m/2}+m/2-6$

From this we get $D_{hx}(2k) = 2^k + k - 6$ and $D_{hx}(2k + 1) = 3 \times 2^{k-1} + k - 6$, for $k \geq 4$.

By comparison, the propagation delay for the 8×9 parallelogram and rectangular layouts of [6] are $D_{[6]}(2k) = (1.1875)2^k - 4$ and $D_{[6]}(2k + 1) = (3.5)2^{k-1} - 4$. Dividing the coefficients of the dominant (2^k) part, we find that (asymptotically) signal propagation is 16.67 percent–18.76 percent faster with the new schemes (the two factors are, respectively, for odd and even tree levels).

The delay for the H-tree in square arrays, as given in [6], is $D_H(2k) = (1.5)2^k - 2$ and $D_H(2k + 1) = 4 \times 2^{k-1} - 1$. By comparison, the delays for the 3×3, 5×7, and 8×8 schemes are, respectively, $D_3(2k) = (1.125)2^k - 1$, $D_3(2k + 1) = 3 \times 2^{k-1} - 1$; $D_5(2k) = (1.0625)2^k - 1$, $D_5(2k + 1) = 3 \times 2^{k-1} - 1$; $D_8(2k) = 2^k$, $D_8(2k + 1) = 3 \times 2^{k-1}$. These times are obtained by solving recurrence relations similar to the hexagonal case. Thus, the new (8×8) scheme achieves an increase in the propagation speed of 33 percent–50 percent (for odd–even tree levels).

The delay times of all the methods are summarized in Table II. Note that the new scheme for square arrays even improves on the methods of [6] for hexagonal arrays.

We note here that in the 8×8 scheme, a tree of level $2k$ occupies a tile of $2^k + 1$ PE's to a side. In such a tile, the shortest possible distance from the center PE to a corner PE is exactly 2^k links, which is the same as the propagation delay of our scheme. Thus, no other scheme occupying the same square of PE's, with root at the center and at least one corner PE as a node, can improve on $D_8(2k)$. In particular, no scheme based on a tile construction (as in Fig. 7) can improve on $D_8(2k)$.

C. Geometric Lower Bounds on Propagation Delay

Define the *distance* between two PE's as the minimal number of links in a path between them. A disk of radius r ($r \geq 0$ and an integer) is the set of all PE's whose distance from a given PE (the center) is $\leq r$. The number of PE's in a disk of radius r is $2r^2 + 2r + 1$ for square arrays and $3r^2 + 3r + 1$ for hexagonal arrays.

In order to embed a tree of level m in a disk of radius r, we must have $2^m - 1 \leq 2r^2 + 2r + 1$ and $2^m - 1 \leq 3r^2 + 3r + 1$ for square and hexagonal arrays, respectively. Solving for r, we find that $r \geq (1/\sqrt{2})2^{m/2} \approx (0.7071)2^{m/2}$ in the square case and (asymptotically) $r \geq (1/\sqrt{3})2^{m/2} \approx (0.57735)2^{m/2}$ in the hexagonal case. Thus, $(0.7071)2^{m/2}$ and $(0.57735)2^{m/2}$ are lower bounds on the propagation delay for the two array types, based only on geometric considerations.

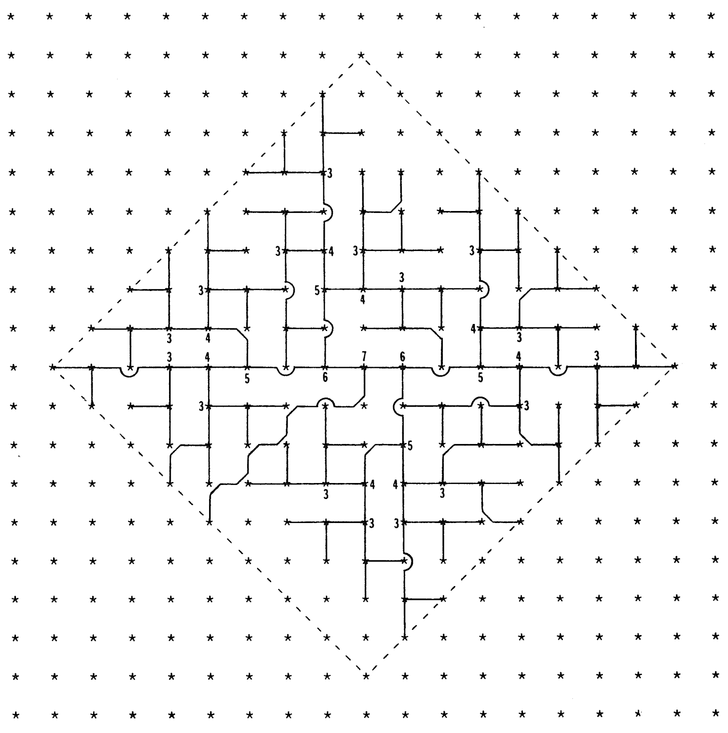
VII. CONCLUSIONS AND OPEN PROBLEMS

By allowing a PE to act both as a node and as a connector we have achieved a much improved utilization of the PE's. The main idea behind our schemes is that global (or long-distance) communications are enabled through local (interprocessor) communications at little or no cost in the processing capability. There are several directions in which these concepts can be explored further.

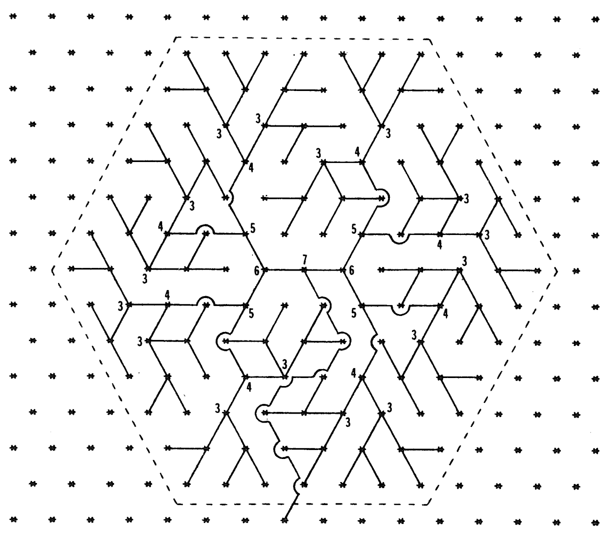
Our embedding schemes were aimed at improving the utilization ratio, and as a byproduct, also improved the propagation delay. From the geometric lower bounds on the delay, we see that there is still room for improvement, perhaps at the cost of less efficient utilization. One such attempt can be seen in Fig. 12, where a binary tree is embedded in a disk of a given radius (in square and hexagonal arrays). Providing a general scheme for such layouts for any tree level could be an interesting problem. Note that it does not help to take such a layout and use it as a basic tile to be replicated, because the delay is determined (asymptotically) by the paths around the basic tiles. It can be shown that for square arrays, the delay would be $D(n) = (\sqrt{2})2^{n/2} - r$ (n even), where r is the radius of a diamond-shaped basic tile.

The lower bounds on the delay given in Section VI can be improved by certain combinatorial considerations. For example, there are only two paths leading out of the root, so some of the PE's on the circumference cannot be be reached from the root by a path of length equal to the radius. A related problem is the following: given a tree level, what is the complexity of determining an embedding of minimal delay?

All the embedding problems can also be considered when connections are allowed to intersect inside a PE. This can be justified by the fact that communications can take place in different layers. In square arrays it would mean that a PE can cross link its neighbors, and in hexagonal arrays it would also



(a)



(b)

Fig. 12. Trees of level 7 embedded in disks. (a) Disk of radius 8 in a square array. (b) Disk of radius 6 in an hexagonal array.

mean that a connection can cross a father–son link inside the PE.

Our "enhanced" PE cannot improve area utilization in the big-O sense, because any embedding in an array of such PE's can be transformed to "simple" PE's with 9 times as may PE's. This is done by a canonical replacement of each enhanced PE with a 3 × 3 square of simple ones—an example is shown in Fig. 13. Similar replacements can also be done on hexagonal arrays. Our technique is thus potentially useful for graphs for which the optimal embedding—to within a constant factor—is already known. Examples of such graphs are arbitrary trees, outerplanar graphs, X-trees, k-dimensional

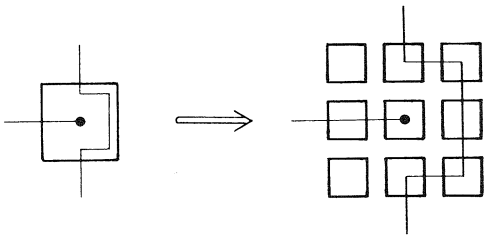


Fig. 13. Canonical transformation of a PE acting as a node and a connector to a 3 × 3 square of PE's, each acting only either as a node or as a connector.

meshes, the shuffle-exchange network and the cube-connected cycles—see [14], [27] for more details and references.

References

[1] J. Bentley and H. T. Kung, "A tree machine for searching problems," in *Proc. 1979 IEEE Int. Conf. Parallel Processing,* Aug. 1979, pp. 257-266.

[2] S. N. Bhatt and C. E. Leiserson, "How to assemble tree machines," in *Proc. 14th ACM Symp. Theory Comput.,* 1982, pp. 77-84.

[3] N. Blum, "An area-maximum edge length trade-off for VLSI layout," *Inform. Contr.,* vol. 65, pp. 45-52, 1982.

[4] R. P. Brent and H. T. Kung, "On the area of binary tree layouts," *Inform. Proc. Lett.,* vol. 11, pp. 46-48, 1980.

[5] B. Chazelle and L. Monier, "A model of computation for VLSI with related complexity results," in *Proc. 13th ACM Symp. Theory Comput.,* 1981.

[6] D. Gordon, I. Koren, and G. M. Silberman, "Embedding tree structures in VLSI hexagonal arrays," *IEEE Trans. Comput.,* vol. C-33, no. 1, pp. 104-107, 1984.

[7] ——, "Restructuring hexagonal arrays of processors in the presence of faults," *J. VLSI Comput. Syst.,* to be published.

[8] E. Horowitz and A. Zorat, "The binary tree as an interconnection network: Applications to multiprocessor systems and VLSI," *IEEE Trans. Comput.,* vol. C-30, pp. 247-253, 1981.

[9] I. Koren, "A reconfigurable and fault-tolerant VLSI multiprocessor array," in *Proc. 8th Annu. Symp. Comput. Architecture,* May 1981, pp. 425-442.

[10] I. Koren and M. A. Breuer, "On area and yield considerations for fault-tolerant VLSI processor arrays," *IEEE Trans. Comput.,* vol. C-33, no. 1, pp. 21-27, 1984.

[11] M. S. Krishnan, "A layout design methodology for VLSI circuits," Ph.D. dissertation, Dept. Elec. Eng., Univ. Southern California, May 1984.

[12] F. T. Leighton and C. E. Leiserson, "Wafer-scale integration of systolic arrays," *IEEE Trans. Comput.,* vol. C-34, no. 5, pp. 448-461, 1985; also in *Proc. 23rd IEEE Symp. Foundations Comput. Sci.* 1982, pp. 297-311.

[13] C. E. Leiserson, "Systolic priority queues," in *Proc. Caltech Conf. VLSI,* Jan. 1979, pp. 199-214.

[14] ——, *Area-Efficient VLSI Computation.* Cambridge, MA: MIT Press, 1983.

[15] C. Mead and L. Conway, *Introduction to VLSI Systems.* Reading, MA: Addison-Wesley, 1980.

[16] C. Mead and M. Rem, "Cost and performance of VLSI computing structures," *IEEE J. Solid State Circuits,* vol. SC-14, no. 2, 1979.

[17] Z. Miller and J. B. Orlin, "NP-completeness for minimizing maximum edge length for grid embeddings," *J. Algorithms,* vol. 6, pp. 10-16, 1985.

[18] A. Mukhopadhyay and R. K. Guha, "Embedding a tree in the nearest neighbor array," in *Proc. 1981 IEEE Int. Conf. Parallel Processing,* pp. 261-263.

[19] R. Negrini, M. Sami, and R. Stefanelli, "Fault-tolerance techniques for array structures used in supercomputing," *Computer,* vol. 19, no. 2, pp. 78-87, 1986.

[20] D. A. Patterson, E. S. Fehr, and C. H. Sequin, "Design considerations for the VLSI processor of X-tree," in *Proc. 6th Annu. Symp. Comput. Architecture,* Apr. 1979, pp. 90-101.

[21] M. S. Patterson, W. L. Ruzzo, and L. Snyder, "Bounds on minimax edge length for complete binary trees," in *Proc. 13th ACM Symp. Theory Comput.,* 1981, pp. 293-299.

[22] A. L. Rosenberg, "The Diogenes approach to testable fault-tolerant arrays of processors," *IEEE Trans. Comput.,* vol. C-32, no. 10, pp. 902-910, 1983.

[23] W. L. Ruzzo and L. Snyder, "Minimum edge length planar embeddings of trees," in *VLSI Systems and Computation,* H. T. Kung, R., Sproull, and G. Steele, Eds. Rockville, MD: Computer Science Press, 1981, pp. 119-123.

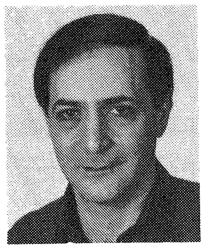
[24] Y. Shiloach, "Linear and planar arrangements of graphs," Ph.D. dissertation, Dep. Appl. Math., Weizmann Instit. Sci., Rehovot, Israel, 1976.

[25] L. Snyder, "Introduction to the configurable highly parallel computer," *Computer,* vol. 15, no. 1, pp. 47-56, 1982.

[26] S. W. Song, "A highly concurrent tree machine for database applications," in *Proc. 1980 IEEE Int. Conf. Parallel Processing,* Aug. 1980, pp. 259-268.

[27] J. D. Ullman, *Computational Aspects of VLSI.* Rockville, MD: Computer Science Press, 1984.

[28] L. G. Valiant, "Universality considerations in VLSI circuits," *IEEE Trans. Comput.,* vol. C-30, no. 2, pp. 135-140, 1981.



Dan Gordon received the B.Sc. and M.Sc. degrees in mathematics from the Hebrew University of Jerusalem, Israel, and the D.Sc. degree in mathematics from the Technion—Israel Institute of Technology, Haifa.

He is Senior Lecturer of computer science at the University of Haifa, Israel. He had previously taught at Purdue University, the University of Cincinnati, and Tel-Aviv University. His current research interests include design and analysis of data structures and algorithms, VLSI, computational geometry, and computer graphics.

Dr. Gordon is a member of the European Association for Theoretical Computer Science (EATCS).