Efficient Planar Embedding of Trees for VLSI Layouts*

NARENDRA AHUJA

Coordinated Science Laboratory, University of Illinois, 1101 W. Springfield Avenue, Urbana, Illinois 61801

Received July 24, 1985; accepted January 28, 1986

The use of planar polygonal partitioning schemes for planar embedding of tree structures is examined. Two layout designs based on recursive square and triangular decompositions are described for trees having branching factors of k^2 and $k^2 - 1$, where k is an integer. The former design allocates the same amount of area to nodes at all levels and requires a total area linear in the number N of nodes in the tree. The latter design assigns increasing area to nodes closer to the root and requires a total area of $O[N(k^2/(k^2 - 1))\log N]$. The longest interconnection has a length of $O(\sqrt{\text{Area}(N)})$ in each case. © 1986 Academic Press, Inc.

1. INTRODUCTION

This paper is concerned with the problem of generating a planar embedding of tree structures. Given a tree we want to create a partition of the chip area such that each node and edge of the tree is assigned to a unique region in the partition. The circuitry necessary to perform the required computations at a node can then be built into the node's region on the chip. Similarly, the regions assigned to the tree edges can house the interconnections between nodes. The interconnections are not allowed to cross.

Possible schemes for generating planar partitions are examined, and two layout designs for trees having branching factors of k^2, where k is an integer and all nodes require regions of the same size, are presented. In most applications where trees are used merely as a data structure, the same amount of hardware may be necessary at each node and hence the corresponding regions may have a fixed area. This is the case addressed in the literature [8]. However, the tree may also be used as an interconnection network for processors in a multiprocessing environment [2, 3]. Here the hardware requirements of nodes may change in different parts of the tree. For example, the nodes closer to the root may need more hardware, and hence, larger regions [2, 3]. The exact rate of increase in the required area will depend upon the nature of computations to be performed. Two layout designs for trees such that the areas of regions assigned to nodes at successive levels increase by a factor of k^2, where k is an integer and the trees have branching factors given by $k^2 - 1$, are presented. For the cases of both constant and variable area per node the two layout designs given differ in type of partitioning scheme used. One of the layouts, that uses the square tessellation for decomposition is more practical than the other that uses the triangular tessellation.

The total area required by the layout of a tree containing N nodes is linear in N in the former case. The longest interconnection has a length which is linearly proportional to the square root of the area. For the latter case, the k^2 – fold increase in the node size with level makes the area grow as $C^{\log N}$ with N. The

*The support of the National Science Foundation under grant ECS8352408 is gratefully acknowledged.

0734-189X/86 $3.00
Copyright © 1986 by Academic Press, Inc.

increase in the length of the longest interconnection is still linear in the square root of the area.

Our motivation for the work described here comes from the prevalence of tree structures in computer vision and image analysis. Trees with branching factor of 4 are used both as data structures [12] and as multiprocessor networks for 2-dimensional image analysis [2]. Trees with branching factor of 8 are useful for 3-dimensional object representation [13] e.g., for collision avoidance [1].

Section 2 describes the details of the problem at hand. Section 3 examines the possible polygonal planar partitioning schemes for generating the regions. Section 4 describes the layouts, and estimates the amount of area occupied by them and the length of the longest interconnection. Section 5 presents concluding remarks.

2. THE PROBLEM

Given a tree, we want to generate a planar layout of nodes interconnected as in the tree. Appropriate circuits can then be built into the regions assigned to the nodes and the interconnections using VLSI techniques. There are two different aspects to this problem.

The first concerns the topology of the tree and its realization as a planar network. It requires a planar embedding scheme that partitions the plane into cells and establishes a node to cell mapping such that the cells can be interconnected as in the tree. The partition should be infinitely repetitive (and divisible) to capture the recursive nature of the interconnections and an arbitrary size of the tree.

The second aspect of the problem concerns the geometry of regions assigned to nodes and to their interconnections. On the one hand, a dense packing of circuits at the nodes and along interconnections is desirable to minimize the chip size. However, this makes the operation of the overall design very complex at high speeds. In addition to the desired, explicitly provided interconnections among nodes there may be electromagnetic interactions among them due to short intervening distances. The interconnections between nodes may not act simply as wires with negligible communication time. Their resistive and stray capacitive effects may lead to a network which is effectively a more complex interconnection of nodes than the desired tree. The deviation in the performance of the interconnections from ideal wires is a problem of major concern in VLSI design. Several models of the dependence of transmission time across an interconnection on its length have been described [6]. Included among these models are those that assume zero, linear, and quadratic increase in transmission time with the length of the interconnection. Different models apply to different technologies and design details. the embedding scheme should therefore minimize the interconnection lengths, which suggests that the interconnected nodes should be assigned to regions close to each other.

In addition, for regularity in mask generation it is desirable that the region interrelationships be identical in different parts of the layout as is the case for the tree, except, of course, for scale effects. This suggests that all node regions should have identical shapes. In conjunction with the earlier observations that interconnected regions be nearby and that the partition be infinitely repetitive, this suggests a recursive decomposition of the plane as the partitioning scheme to obtain the regions. "Recursiveness" here means that the node regions are recursively partitioned to house interconnected nodes. The requirement of infinitive repetitiveness is met by ensuring that the regions can be split (merged) to decrease (increase) in size as much as necessary without altering their shape. Finally, for VLSI implementation

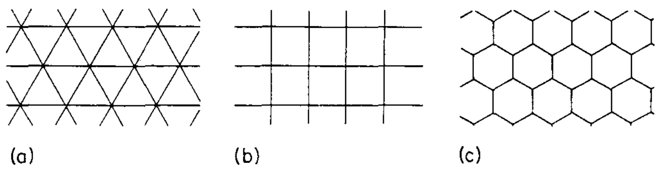

FIG. 1. Regular tessellations: (a) triangular, (b) square, and (c) hexagonal.

it is desirable that the partition be defined by straight line segments having a limited number of orientations. We will now discuss possible polygonal partitioning schemes and examine their suitability for embedding trees in the light of the requirements stated above.

3. PARTITIONING SCHEMES

A polygonal partition is defined by a collection of polygonal faces (cells). Let k denote the number of sides of a cell in a given partition. Let v denote the number of cells meeting at a vertex. It can be shown [11, 7] that there exist only three partitions of the plane in which the value of $k(v)$ is same for all cells (vertices)—the possible (k, v) values being $(3, 6)$, $(4, 4)$, and $(6, 3)$. We call such tessellations kv regular tessellations [4]. They correspond to a division of the plane into regular triangular, square (rectangular), and hexagonal cells, respectively (Fig. 1). The triangular and the hexagonal tessellations form a pair of dual graphs.

If the value of k is allowed to vary from cell to cell, keeping v fixed, eight additional types of partitions are also possible [7]. By definition, these partitions do not consist of congruent cells, but a mixture of as many different regular k-gons as there are values of k. They are called semiregular (Archimedean) tessellations. A semiregular tessellation may be characterized by an ordered sequence of v integers, where the ith integer denotes the number of sides of the ith cell surrounding a vertex, starting at any of the surrounding cells arbitrarily, and moving, say, clockwise. In this notation, the eight semiregular tessellations are given by $(3, 12, 12)$, $(4, 6, 12)$, $(4, 8, 8)$, $(3, 6, 3, 6)$, $(3, 4, 6, 4)$, $(3, 3, 3, 3, 6)$, $(3, 3, 3, 4, 4)$, and $(3, 3, 4, 3, 4)$ (Fig. 2).

Both regular and semiregular tessellations possess the infinite repetitiveness property. We now examine them with respect to the requirement of recursive decomposability. Cells in a regular tessellation are all congruent. If we can partition each cell further into smaller cells such that the new tessellation still is a kv regular tessellation, then the infinite decomposability requirement is met. Alternatively, it should be possible to merge cells locally to obtain a kv regular tessellation with larger cells.

Clearly, the triangular and square tessellations possess this property (Fig. 3). On the other hand, cells in a hexagonal tessellation cannot be further divided into regular congruent hexagons. To prove this, imagine merging neighboring hexagons (of side d) in a regular tessellation to form a larger hexagon. By the requirement of recursive decomposability, the edges of the larger hexagon must be contained in the given tessellation. However, all the straight line segments in the tessellation are of length d; they cannot possibly define hexagons of side longer than d. A similar argument rules out all semiregular tessellations except for $(3, 6, 3, 6)$ and $(3, 3, 3, 3, 6)$. In the latter two, the placement of star-shaped cells [5] leaves holes (Fig. 4). Thus

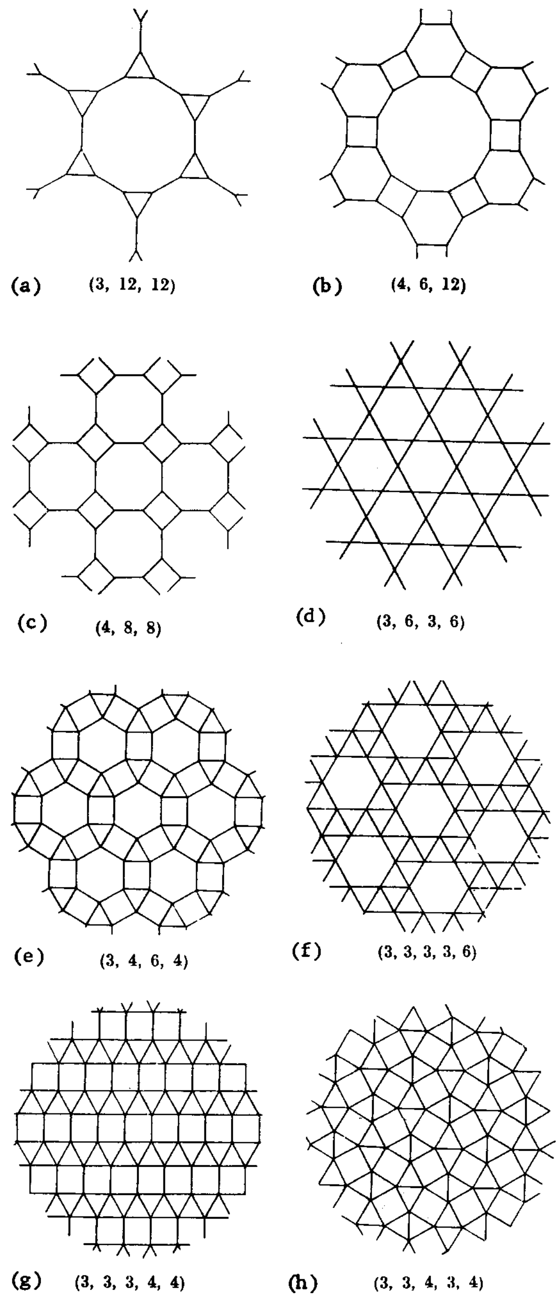

FIG. 2. Semiregular tessellations [7].

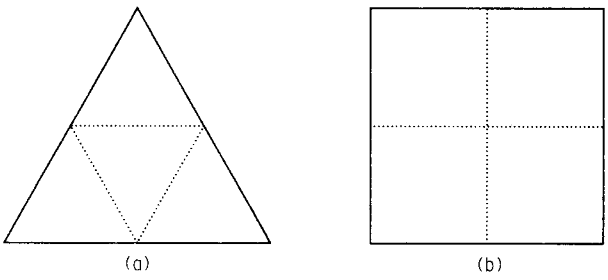

FIG. 3. Cells (dotted lines) in (a) triangular and (b) square, tessellations merge into larger cells (continuous lines).

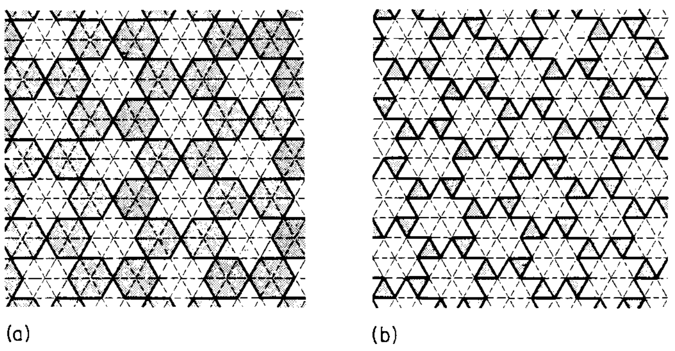

FIG. 4. The placement of the star-shaped tile in the semiregular tessellations (a) (3,6,3,6) and (b) (3,3,3,3,6), leaves holes (hatched) between adjacent tiles.

adjacent cells cannot merge to form a larger cell, making the tessellation recursively nondecomposable. The regular square and triangular tessellations, therefore, are the only partitioning schemes that place no restriction on region size.

If an upper limit on the cell dimensions, and therefore, on the largest allowed region size is acceptable, then some of the remaining semiregular tessellations could also meet the recursive decomposability requirement. For example, cells in those tessellations involving only squares and triangles (Fig. 2g, h) may be recursively partitioned using regular square and triangular decompositions. One must start with the largest required cell sizes, since cells do not combine into larger cells. However, the requirement that all cells have a fixed shape rules out such choices.

Thus, we have the square and triangular tessellations as two possible choices for an acceptable partitioning scheme. However, the triangular tessellation has two disadvantages: first, it involves a grid containing lines oriented in three different directions compared to two for the square case, and second, to make the best use of space may often require that the chips be triangular in shape. In what follows, we will consider both partitioning schemes to design two sets of layouts, although the square tessellation based layouts may be more appropriate for reasons given above.

4. LAYOUTS

The decomposition of a cell, say a square, into smaller squares in the planar decomposition corresponds to branching of a node in the tree. While the polygonal

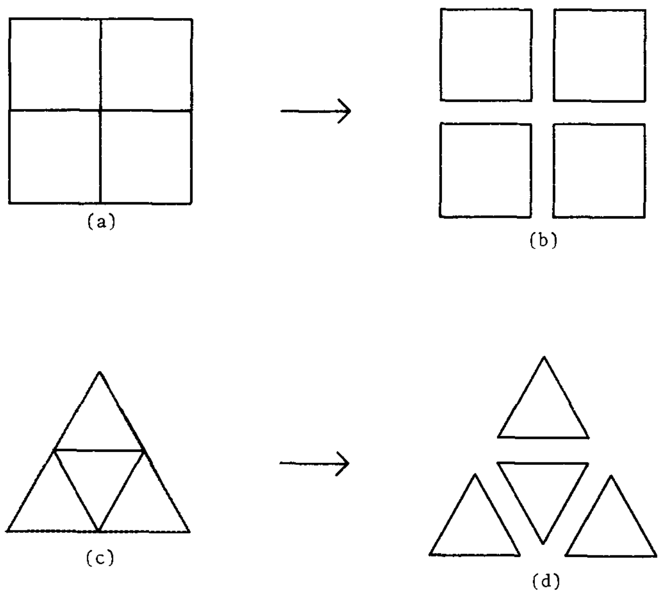

FIG. 5. The decomposed square cell in (a) is enlarged in (b) to create space for interconnections; (c) and (d) are analogous to (a) and (b), but for the triangular cell.

decomposition as described in Section 3 yields appropriate space for the nodes, there is no space allocated for interconnections, or the edges of the tree. There are two important issues to be resolved before a layout can be designed:

(1) How are the internode connections to be implemented, i.e., how is the decomposition scheme to be modified to allocate space for the necessary connections between nodes?

(2) What is the relationship between the sizes of regions assigned to a node and its children?

The first problem has a simple solution. The parent cell can be enlarged while retaining the sizes of the subcells, and creating narrow strips of vacated space between adjacent subcells. The strips are made sufficiently wide so as to accommodate connections between nodes which are routed through the strips. Figure 5 illustrates this for both square and triangular cases. In fact, if the width of the basic cell is sufficient to hold interconnections then some of the subcells can touch each other. For example, in the repetitive pattern in Fig. 6a, the central square cells can touch the outer cells at corners. If e denotes the edge-length of the basic square cell, then the edge-length of the region occupied by a two-level tree will be $3e$, corresponding to an area of $9e^2$. With each additional level the area will grow by a factor of 9. Of course, only a fixed number of intersections are required to pass through each strip, irrespective of the number of levels in the tree.

The answer to the second question depends upon the specific application of the tree. When the tree is used only for its topological structure, all the nodes being identical as is the case for many data representation applications, equal size regions can be used for all nodes. Figure 6a illustrates for this case the basic repetitive

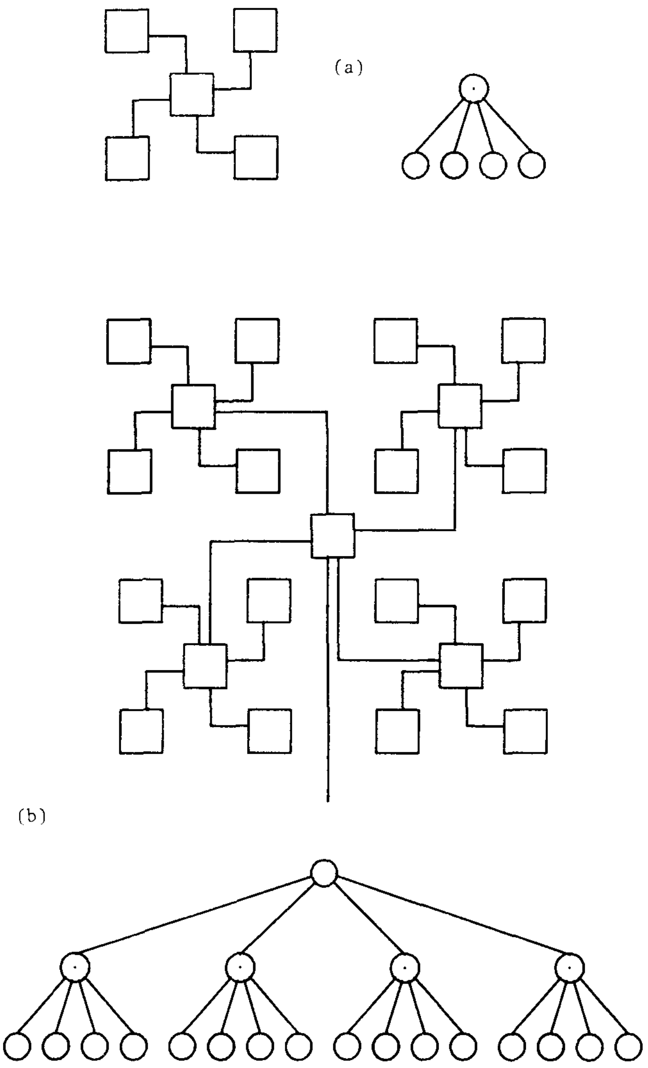

FIG. 6. (a) The basic repetitive pattern for a quadtree ($k = 2$, $B = k^2$) layout using square tessellation. (b) A three-level layout generated by (a). All nodes are of equal size.

structure required for planar embedding of a quadtree (branching factor $B = 4$) using a square tessellation, where the node corresponding to the newly inserted central region is a parent of the nodes corresponding to the moved subcells. Figure 6b shows how this basic two-level structure can be repeated to generate a larger tree, in this case, consisting of nodes at three levels. Figure 7 is analogous to Fig. 6 but uses the triangular decomposition. If the basic cell has an edge length of t and this is sufficient to house the interconnections, then the repetitive patter of Fig. 7a may have an edge length of only $3t$. The ara occupied by a tree grows by a factor of 9 with every level as is the case for the square tessellation. Thus, both square and triangular tessellations make equally efficient use of space. In certain other applica-

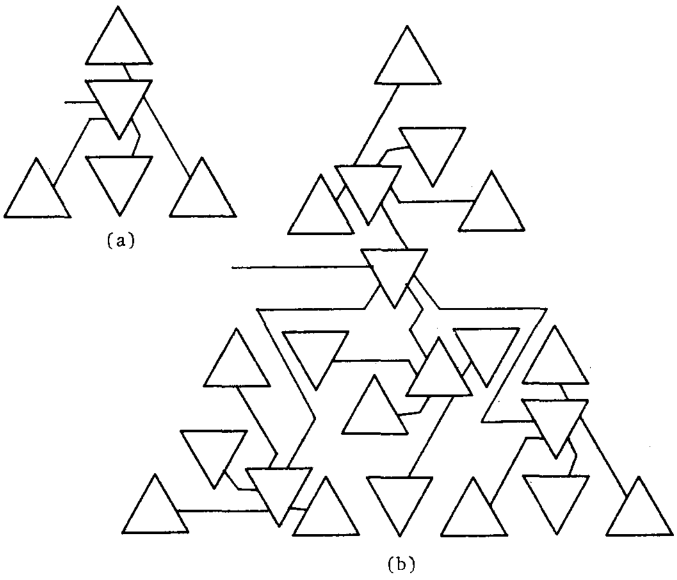

FIG. 7. Analogous to Fig. 6 but using triangular tessellation.

tions of tree structure, the node characteristics depend upon its location in the tree, e.g., in multiprocessor pyramid architectures [2, 3] each node houses a processor whose size increases with the height of the node. This can be implemented by recursively splitting all but one of the regions; the unsplit large region serves as the parent node. Figures 8 and 9 are analogous to Figs. 6 and 7, and illustrate the layouts for a ternary tree ($B = 3$).

4.1. *Permissible Branching Factors*

Our examples so far have involved trees with branching factors of only 4 and 3, corresponding to the cases of equal and variable size nodes, respectively. In the former case the branching factor is the same as the number of subcells obtained by

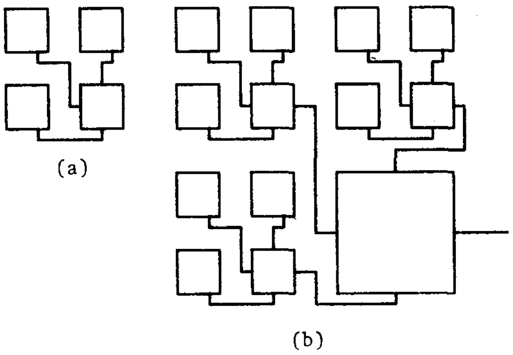

FIG. 8. Analogous to Fig. 6 but for the case when node size increases (quadruples) with level resulting in a ternary tree ($k = 2$, $B = k^2 - 1$).

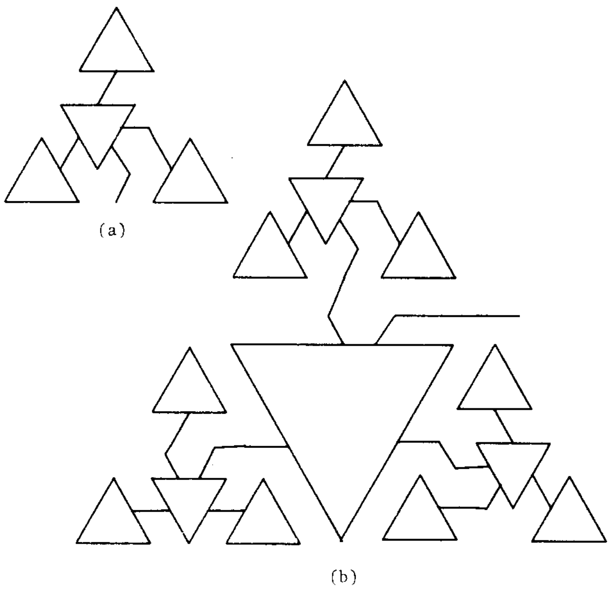

FIG. 9. Analogous to Fig. 8 but using triangular tessellation.

splitting a cell, and in the latter case it is one less since one of the subcells is used by the parent node.

The decomposition scheme of Fig. 5 that gives rise to the above cases ($B = 3, 4$) can be described as follows. Divide each side of the parent cell into two parts ($k = 2$). From each midpoint draw a line parallel to each of the remaining sides of the cell to obtain subcells. Since each dimension of a subcell is $1/2$ of the larger cell, its area is $1/22 = \frac{1}{4}$ that of the larger cell. Thus, there are four subcells per cell giving $B = 3, 4$ for the two cases of node sizes. In general, if each side of a cell is divided into k equal segments and lines parallel to the remaining sides are drawn from the endpoints of the segments, k^2 subcells will result. Thus, trees with $B = k^2$ or $k^2 - 1$ could be laid out. Figures 10 and 11 illustrate layouts for $k = 3$ and $B = k^2$. The layouts for the case of $k = 3$ and unequal node sizes ($B = k^2 - 1$) are shown in Figs. 12 and 13.

In all the layouts the line segments are drawn along an underlying (square or triangular) grid tessellation, not shown in the figures. The cell decomposition for the $B = k^2 - 1$ case is the same as that shown in Fig. 5. However, to obtain the $B = k^2$ case, a new parent node is inserted as the ($k^2 + 1$)th node in the decomposition given in Fig. 5. The locations of the k^2 subcells in Fig. 5 are perturbed to accommodate the new subcell as near the cell center as possible while minimizing the loss in packing density.

Note that if the 4-way ($k = 2$) square decomposition of a region shown in Fig. 5a is carried out in two steps, first into two rectangles and then each rectangle into two squares (Fig. 14), the resulting scheme gives, as a special case, the standard H-layout for binary trees proposed in the literature [9]. Such cascading of steps is not possible for the triangular case.

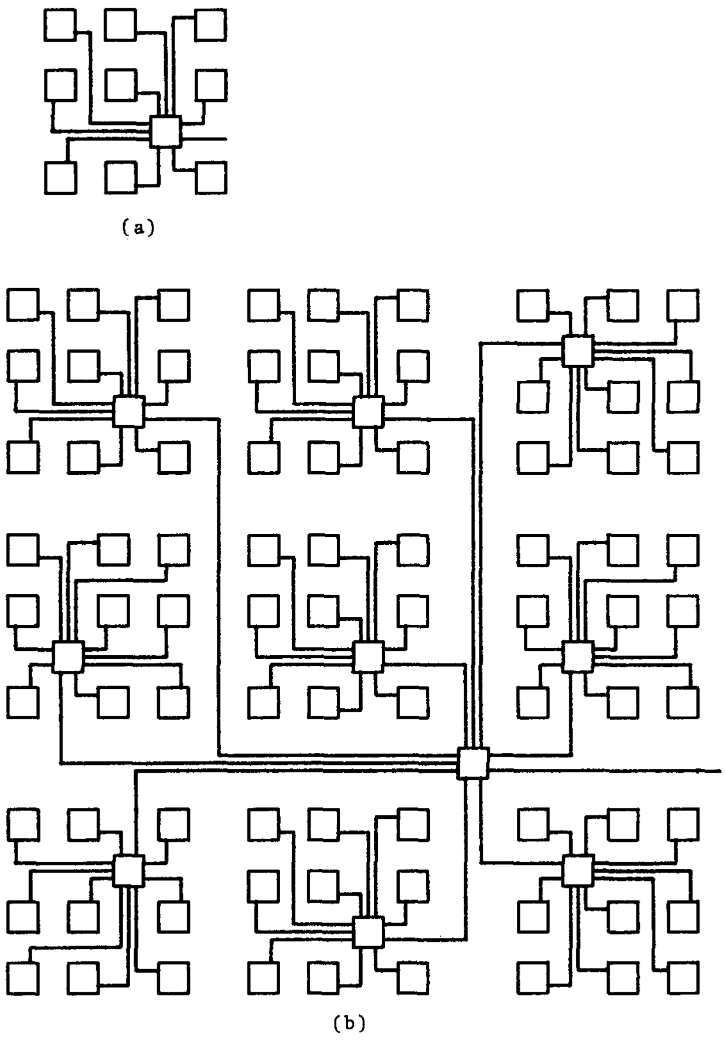

FIG. 10. (a) The basic repetitive pattern and (b) the layout for a three-level tree ($k = 3$, $B = k^2$) using square tessellation. All nodes are of equal size.

4.2. *Area and Interconnection Length*

Each successive step splits a node region into children node subregions and the interregion spaces of small width ($\ll 1$) for interconnections. Both the square and triangular decomposition schemes use the same amount of area per node. Let d denote the edge length of the layout and let N denote the number of nodes in the tree at levels $0, 1, \ldots, L$. We will derive the relationship between the total area of the layout and the number of nodes in the tree, for the square tessellation and for each of the two cases $B = k^2$ and $B = k^2 - 1$. The area of the interconnection regions will be ignored in computing the layout area, since the width of such regions is

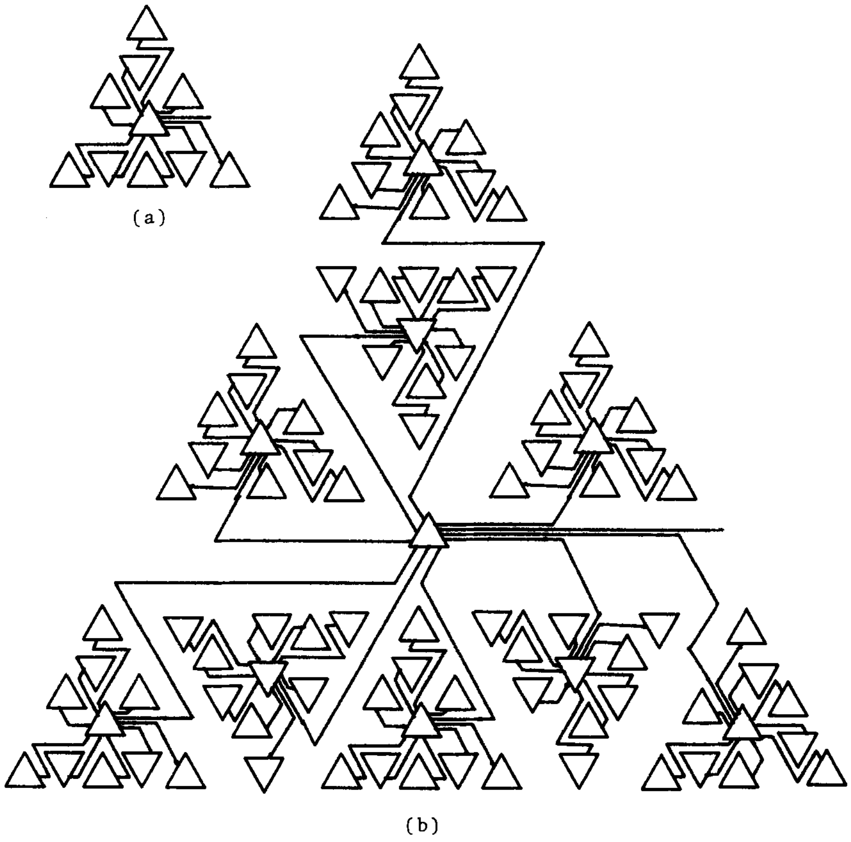

FIG. 11. Analogous to Fig. 10 but using triangular tessellation.

assumed to be small ($\ll 1$), and their length grows linearly with the square root of the area.

Case 1. $B = k^2$. Each step of the decomposition divides the layout edge into $k+1$ segments: k equal length segments occupied by k children nodes and a unit length segment corresponding to the parent node (see Figs. 6, 10). If the tree has L levels then all segments along the layout edge will be of unit length after L steps of recursive decomposition. Thus L and d are related:

$$d = 1 + k + k^2 + \cdots + k^L = \frac{k^{L+1} - 1}{k - 1} \approx \frac{k^{L+1}}{k - 1}.$$

Now,

$$N = 1 + k^2 + (k^2)^2 + \cdots + (k^2)^L = \frac{(k^2)^{L+1} - 1}{k^2 - 1} \approx \frac{k^{2(L+1)}}{k^2 - 1}.$$

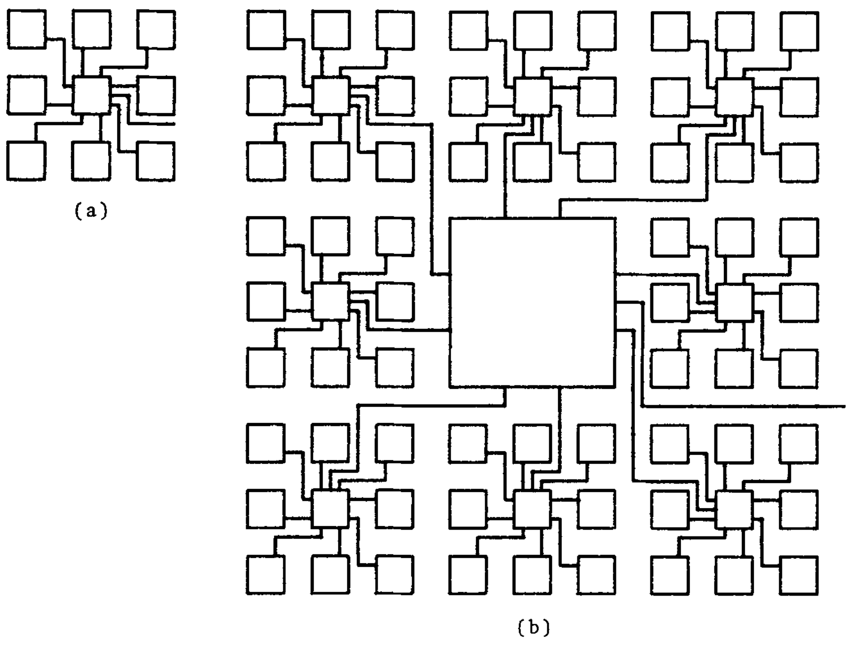

FIG. 12. Analogous to Fig. 8 but for $k = 3$.

The layout area per node is given by

$$\frac{\text{Area}(N)}{N} = \frac{d^2}{N} = \frac{k^{2(L+1)}}{(k-1)^2}\frac{k^2-1}{k^{2(L+1)}} = \frac{k+1}{k-1}$$

$$\text{Area}(N) = \frac{k+1}{k-1}N.$$

Thus, the area Area(N) of the layout of a tree consisting of N nodes is linear in N,

$$\text{Area}(N) = O(N).$$

Case 2. $B = k^2 - 1$. Each step of the decomposition now divides the layout edge into k equal segments (see Figs. 8, 12). After L steps of recursive decomposition the layout edge will consist of unit length segments. Thus,

$$d = k^L.$$

Now,

$$N = 1 + (k^2-1) + (k^2-1)^2 + \cdots + (k^2-1)^L$$
$$= \frac{(k^2-1)^{L+1} - 1}{K^2-2} \approx \frac{(k^2-1)^{L+1}}{k^2-2}.$$

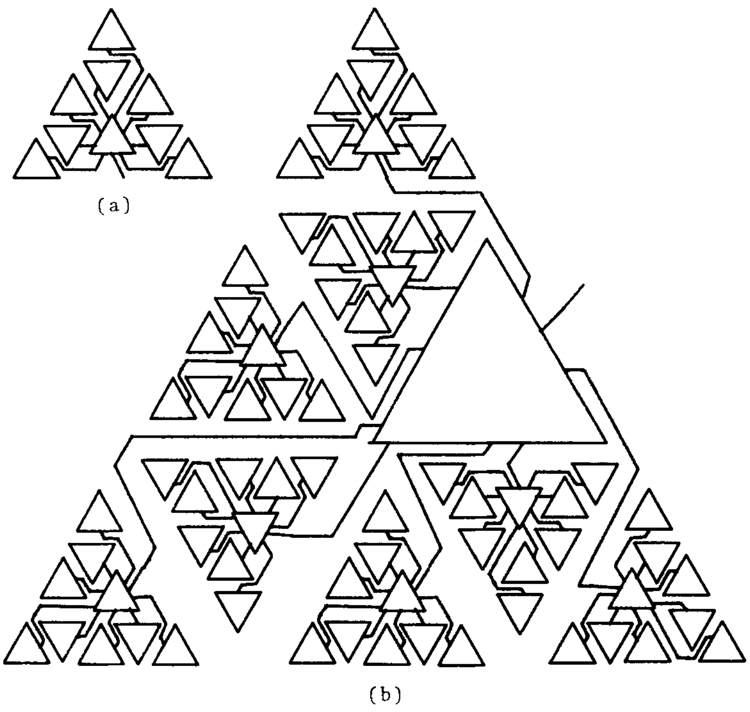

FIG. 13. Analogous to Fig. 8 but using $k = 3$ and triangular tessellation.

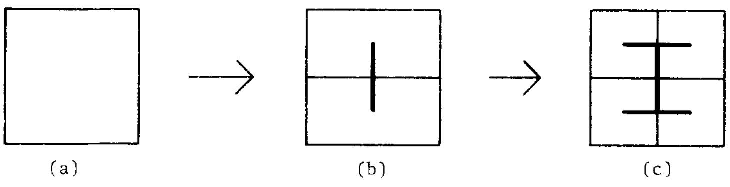

FIG. 14. A two step decomposition (b), (c) of a square (a), similar to Fig. 5a, yields the H-tree layout of a binary tree proposed in literature [9].

Alternately,

$$(k^2 - 1)^{L+1} \approx (k^2 - 2)N$$
$$L + 1 \approx \log_{(k^2-1)}(k^2 - 2) + \log_{(k^2-1)}N$$
$$L = \log_{(k^2-1)}N + \left(\log_{(k^2-1)}(k^2 - 2) - 1\right).$$

Thus,

$$L \approx \log_{(k^2-1)}N.$$

The layout area per node is given by

$$\frac{\text{Area}(N)}{N} = \frac{d^2}{N} = \frac{k^{2L}(k^2 - 2)}{(k^2 - 1)^{L+1}}$$

$$= \frac{k^{2L}}{(k^2 - 1)^L} - \frac{1}{(k^2 - 1)^{L+1}}$$

$$\approx \left(\frac{k^2}{k^2 - 1}\right)^L.$$

Thus,

$$\text{Area}(N) = N\left(\frac{k^2}{k^2 - 1}\right)^{\log_{(k^2-1)} N}$$

Here the layout area grows faster with the number of nodes than is the case for $B = k^2$. This is to be expected as the same amount of layout area holds fewer nodes $(k^2 - 1 = B < k^2)$, the nodes at higher levels occupying larger areas. However, if $k^2 \gg 1$ then the number of leaves, and hence the area occupied by them, will dominate the total layout area. Thus the dependence will be almost linear:

$$\frac{k^2}{k^2 - 1} \approx 1$$

and $\text{Area}(N) \approx O(N)$.

The longest interconnection occurs between the root and its children, since all lower levels are obtained by compact polygonal decomposition. Thus, the longest interconnection has a length on the order of the dimension (diameter) of the largest region (see figures). Therefore,

$$\text{Length}(N) = O\left(\sqrt{\text{Area}(N)}\right) \approx O(\sqrt{N}).$$

5. CONCLUSIONS

We have examined possible schemes for partitioning the plane into regions such that the nodes and edges of a tree can be assigned to these regions to obtain a planar embedding of the tree. The interconnections are not allowed to cross. Only two different partitioning schemes based upon the regular square and triangular tessellations are found to permit the desired node to region mapping. For each scheme, two different layout designs have been described. In the first, all nodes are assumed to require a fixed area, for example, when the tree is used as a data structure. A general method for embedding k^2-ary trees has been described that takes $O(N)$ area and has the longest interconnection of length $O(\sqrt{\text{Area}(N)})$, for a tree containing N nodes. In the second case, the nodes at levels closer to the root are assumed to require increasing amount of area, for example, when the tree is used as a processor interconnection network for multiprocessing. A general method for embedding $(k^2 - 1)$-ary trees has been described that takes $O(NC^{\log N})$ area $(C = k^2/(k^2 - 1)$ and log is to the base $(k^2 - 1))$. the longest interconnection has a length $O(\sqrt{\text{Area}(N)})$.

Trees with branching factors other than k^2 (or $k^2 - 1$, when node area varies) do not permit efficient layouts as per the schemes described in the paper. Consequently, such a tree may be realized by using a chip implementing a larger tree but activating only the desired subtree. Thus, for example, a tree with $B = k^2 + r, 0 < r < 2k + 1$, may be realized by using a chip designed for a $(k + 1)^2$-ary tree, but using only that subtree containing edges labelled $1, 2, \ldots, k^2 + \mathrm{r}$ down to the leaves. For the purpose of image analysis, quadtree and octree are two important tree data structures, useful for representing 2- and 3-dimensional images, respectively. Quadtree layouts are realized using $k = 2$ and $B = k^2$. Octrees are realized by setting $k = 3$ and $B = k^2$, and using only eight children of each node. Multiprocessor pyramids for 2-dimensional image analysis [2, 3] that require $B = 4$ and approximately doubling node size with level can not be laid out in linear area using $k = 2$, and $B = k^2 - 1$. However, their approximate 3-dimensional counterparts ($B = 8$), with the node area increasing by a factor of 9 with level, can be laid out using $k = 3$, and in a total layout area $O(N(1.1)^{\log_{(1.1)} N})$.

We have considered layouts that employ explicit interconnections between nodes, unlike the case in "implicit wiring" described in [10], where the connections may be routed through nodes acting partly as switches. Indeed, by allowing implicit wiring, any k-ary tree layout could be used to realize a k^n-ary tree, for any integer n, by treating subtrees of n levels as single nodes. Thus, for example, the H-tree layout for binary trees could be used to realize quadtrees, octrees, or any other 2^n-ary tree. The message switching operations at nodes increase the internode communication time. This factor is absent in the layouts discussed in this paper where the communication time is primarily determined by the length of the interconnection.

REFERENCES

1. N. Ahuja, R. T. Chien, R. Yen, and N. Bridwell, Interference detection and collision avoidance among three dimensional objects, in *Proc. 1st National Conf. on Artificial Intelligence*, Stanford University, August 19–21, 1980, pp. 44–48.
2. N. Ahuja and S. Swamy, Multiprocessor pyramids for bottom-up image analysis, in *Proc. IEEE Conf. on Pattern Recognition and Image Processing*, Las Vegas, June 13–17, 1982, pp. 380–385.
3. N. Ahuja and S. Swamy, Interleaved pyramid architectures for bottom-up image analysis, in *Proc. 6th Int. Conf. on Pattern Recognition*, Munich, Germany, October 19–22, 1982, pp. 388–391.
4. N. Ahuja and B. Schachter, *Pattern Models*, Wiley, New York, 1983.
5. N. Ahuja, On approaches to polygonal decomposition for hierarchical image representation, *Comp. Vision Graphics Image Processing*, November 1983, pp. 200–214.
6. G. Bilardi, M. Pracchi, and F. P. Preparata, A critique of network speed in VLSI models of computation, *IEEE J. Solid State Circuits* **SC-17**, 1982, 696–702.
7. L. Fejes Toth, *Regular Figures*, Macmillan, New York, 1964.
8. C. E. Leiserson, *Area-Efficient Graph Layouts (for VLSI)*, Department of Computer Science Technical Report, Carnegie-Mellon University, August 1979.
9. C. A. Mead and A. Rem, Cost and performance of VLSI computing structure, *IEEE J. Solid State Circuits* **SC-14**, 1979, 455–462.
10. A. Mukhopadhyay and R. K. Guha, Embedding a tree in the nearest neighbor array, in *Proc. Int. Conf. on Parallel Processing*, August 1981, pp. 261–263.
11. O. Ore, *Graphs and Their Uses*, Random House, New York, 1963.
12. A. Rosenfeld, Quadtrees and pyramids for pattern recognition and image processing, in *Proc. 5th Int. Conf. on Pattern Recognition*, 1980, pp. 802–811.
13. S. N. Srihari, Representation of three-dimensional digital images, *ACM Comput. Surveys*, December 1981, 399–424.