AD-A096 367 CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF COMPUTER --ETC F/G 9/5

AREA-EFFICIENT GRAPH LAYOUTS (FOR VLSI).(U)

AUG 80 C E LEISERSON N00014-76-C-0370

UNCLASSIFIED CMU-CS-80-138 NL

1 OF 1

AD A
096367

END

DATE FILMED

4-81

DTIC

AD A096367

LEVEL II

CMU-CS-80-138

12

Area-Efficient Graph Layouts (for VLSI).

Charles E. Leiserson

Department of Computer Science
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

12 27

August 1979
Last Revised 13 August 1980

DTIC
ELECTE
MAR 16 1981
S D
E

DEPARTMENT
of
COMPUTER SCIENCE

Carnegie-Mellon University

81 3 16 105

CMU-CS-80-138

Area-Efficient Graph Layouts (for VLSI)

Charles E. Leiserson

Department of Computer Science
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

August 1979
Last Revised 13 August 1980

Abstract

Minimizing the area of a circuit is an important problem in the domain of *Very Large Scale Integration*. We use a theoretical VLSI model to reduce this problem to one of laying out a graph, where the transistors and wires of the circuit are identified with the vertices and edges of the graph. We give an algorithm that produces VLSI layouts for classes of graphs that have good *separator theorems*. We show in particular that any planar graph of n vertices has an $O(n \lg^2 n)$ area layout and that any tree of n vertices can be laid out in linear area. The algorithm maintains a sparse representation for layouts that is based on the well-known UNION-FIND data structure, and as a result, the running time devoted to management of this representation is nearly linear.

This research was sponsored in part by the Defense Advanced Research Projects Agency (DOD) ARPA Order No. 3597 which is monitored by the Air Force Avionics Laboratory Under Contract F33615-78-C-1551, by the National Science Foundation under Grant MCS 78-236-76, and by the Office of Naval Research under Contract N00014-76-C-0370. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the United States Government. Charles E. Leiserson is also supported by a Fannie and John Hertz Foundation fellowship.

1. Introduction

The remarkable advance of very large scale integrated (VLSI) circuitry has sparked research into the design of algorithms suitable for direct hardware implementation. To the computer theorist, VLSI provides an attractive model of parallel computation for three reasons. First of all, the number of components that can fit on a single chip is large, and beyond that has been doubling every one to two years. It is currently possible to place 10^5 components on a single chip, and it is projected that this number will very likely grow to 10^7 or even 10^8. These large numbers make asymptotic analysis and other theoretical tools applicable to this engineering discipline. Secondly, VLSI hardware expense can be related directly to the very mathematical and geometric cost function of *area*. Unlike older technologies, the components and interconnections between components are made out of the same "stuff" in VLSI, and hence area is a uniform cost measure for both. Finally, VLSI provides a model of parallel computation that includes communication costs as well as operation counts. The cost of communication is represented explicitly as the area of a fixed-width wire between two processors. In fact, interconnections can consume most of the area of an integrated circuit chip. A major goal, therefore, is to minimize the area required by particular interconnection schemes. This paper examines the problem in an abstract setting: *"Given a graph, produce an area-efficient layout."*

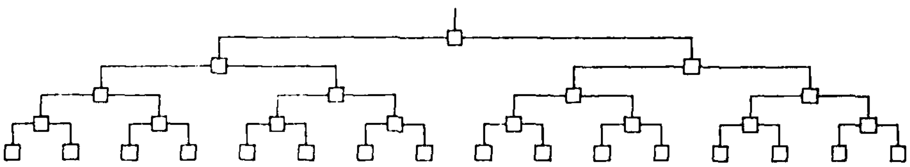

Figure 1: An $O(n \lg n)$ layout of a complete binary tree.

To illustrate the subtleties inherent in this problem, consider laying out a complete binary tree of $n = 2^k-1$ vertices. Figure 1 shows an obvious solution that requires $O(n \lg n)$ area—$O(n)$ across the bottom times $O(\lg n)$ height. Observe that as we ascend the tree from the leaves to the root, the number of wires is halved from one level to the next, but the length of the wires doubles. This means that the amount of wire devoted to each level of the tree is the same. The recurrence that describes the area required by this layout is $A(n) = 1$ for $n = 1$, and

$$A(n) \;=\; 2A(\lfloor n/2 \rfloor) + n/2$$

for $n = 2^k-1$ where $k > 1$.

There is a more efficient solution to this embedding problem. The so-called *H-tree* layout [17] shown in Figure 2 requires only $O(n)$ area in spite of the fact that relatively long wires are used towards the root of the tree. In this layout, the number of wires is halved from level to level as we ascend to the root, but the length of the wires doubles only every two levels. Whereas, the standard $O(n \lg n)$ layout uses just one dimension for routing most of the wires, the H-tree makes better use of both spacial dimensions. The recurrence describing the area required by the H-tree is more complex than the previous one because of its nonlinear form: $A(n) = 1$ for $n = 1$, and

$$A(n) \;=\; 4A(\lfloor n/4 \rfloor) + 4\sqrt{A(\lfloor n/4 \rfloor)} + 1$$

for $n = 2 \cdot 4^k-1$ where $k \geq 1$.

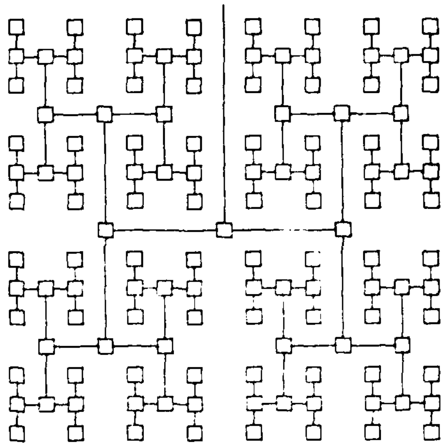

Figure 2: The *H-tree* layout of a complete binary tree.

This recurrence can be solved by taking the square root of both sides of the equation, and rewriting it in terms of $\sqrt{A(n)}$, the length of the edge of the layout. The new recurrence is a simple divide-and-conquer recurrence which has solution $O(\sqrt{n})$ for the edge of the layout.

The remainder of this paper is organized as follows. Sections 2 through 5 contain background material that will be used in later sections. Section 2 contains a formulation of the VLSI layout model, and Section 3 gives the definition of a separator theorem. Some results regarding the areas and *aspect ratios* of layouts are proved in Section 4. In Section 5, an important nonlinear recurrence equation is solved.

Section 6 brings the disjoint ideas of the previous sections together by using separator theorems to help construct VLSI layouts. Many corollaries follow from this main result, and they are explored in Section 7. In Section 8, an implementation of the layout algorithm is presented which is based on the UNION-FIND algorithm analyzed by Tarjan [22]. It is shown that the time required for maintaining the representation of a layout is nearly linear. Section 9 uses many of the techniques from earlier sections to investigate a layout model in which the vertices of a graph are constrained to lie on a line. Finally, Section 10 tries to place the work of this paper in proper perspective.

2. The VLSI Model

Before presenting the VLSI model used in this paper, it is worthwhile to examine some of the attributes of VLSI technologies. VLSI components—wires and transistors—are constrained to lie in layers on a wafer of silicon. Because the number of layers is small (usually under six), the size of a VLSI chip can be measured by the total area of silicon used—the layers contributing to the ability of wires to cross. Every VLSI fabrication process has a natural metric, the *minimum feature size* λ, which is the width of the narrowest wire that can be manufactured.[1] The smallest transistor that can be manufactured is a square with edge λ and area λ^2. Since a wire of length L consumes λL area, it is not unusual for much of the area of a chip to be consumed by wires.

[1] Mead and Conway [[illegible]] in fact define λ to be half the width of the narrowest manufacturable wire.

Intuitively, the VLSI model should make one-to-one correspondences between edges in the graph and wires in the layout, and between vertices in the graph and transistors in the layout. The mapping between edges and wires seems straightforward enough, but there are many issues to be resolved in establishing a correspondence between vertices and transistors. An important one is that a vertex in a graph may have large degree, and yet on an integrated circuit, an arbitrarily large number of wires cannot come together at a single point. There just isn't enough room. Another problem arises from the fact that a transistor occupies area. What assumptions should be made about the size and shape of that area?

In this paper, we resolve these difficulties by restricting the discussion to classes of graphs with vertex degrees that are bounded by a constant, and by further assuming that vertices require only a constant area of silicon. Although these constraints may seem severe at first, the results of the paper are easily generalized to more complex models. For example, there is a simple transformation from an arbitrary graph to a trivalent graph such that each vertex of the original graph is a block of the trivalent graph. Another way that the model can be extended is to allow several transistors to be connected by a single wire. This is easily accomodated by considering bipartite graphs—vertices in one set represent transistors and those in the other represent wires.

Having resolved the graph-theoretic issues, we now turn to the modeling of the layouts u.emselves. The VLSI model proposed here is similar to that of Thompson [23] in which wires have unit width and only a constant number (two) may cross at a point. Vertices are placed on a rectangular grid so that each lies within a grid square. Edges run horizontally and vertically, one per grid square, except that an edge running horizontally may cross one running vertically.[2]

Layouts that are designed with this model have the property that they are *sliceable*. That is, a horizontal or vertical line can be used to bisect the layout, the pieces can be moved apart, and the severed wires can be reconnected to realize the original topology. Slicing can be used to generate new layouts from old ones. For example, Figure 3 shows how slicing enables a new edge to be routed between two existing vertices in a layout. Two horizontal and two vertical cuts are made through the layout to expose the the vertices that are to be connected. (Actually, two slices in one direction and one in the other always suffice.) The pieces are separated by a grid unit, the severed edges are reconnected across the gaps, and a new edge which connects the vertices is run through the gaps. If the length in grid units of the original layout was L and the width W, the new layout has length at most $L+2$ and width at most $W+2$. It should be noticed that the slices through the layout must be straight—a staircase cut may require the pieces to be separated by more than a single grid unit for a new edge to be routed.

3. Separator Theorems

Recently, Lipton and Tarjan [14] showed that any planar graph of n vertices can be divided into two subgraphs of approximately the same size by removing only $O(\sqrt{n})$ vertices. Since the subgraphs are themselves planar, this *separator theorem* provides a basis for exploiting the divide-and-conquer paradigm [1]. We shall find it convenient to alter the definition of separator theorem that Lipton and Tarjan give. Whereas they bisect a graph by removing vertices, we shall remove edges. Since we are principally concerned with classes of graphs with bounded degree, the definition we give is equivalent except for the values of the constants in the definition.

[2] So that wires do not change often from one layer to another, many wire-routing programs use a *Manhattan* scheme [12] in which all horizontally running wires are placed on one layer and all vertically running wires on the another.

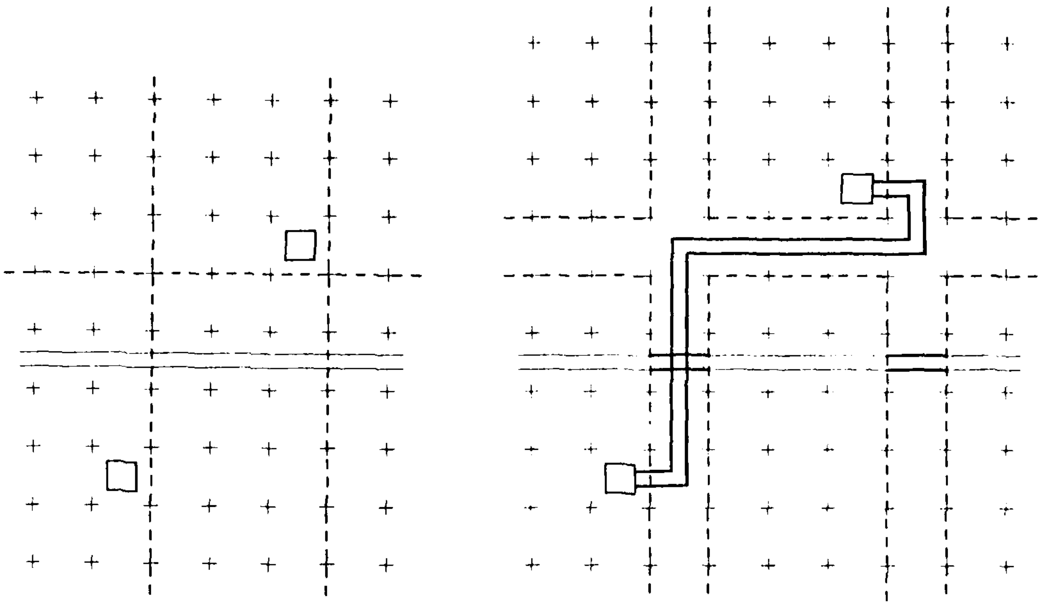

Figure 3: Two horizontal and two vertical slices are more than sufficient to route an edge.

Definition: Let S be a class of graphs closed under the subgraph relation, that is, if G is an element of S, and G' is a subgraph of G, then G' is also an element of S. An $f(n)$-*separator theorem* for S is a theorem of the following form.

> There exist constants α_S and c_S where $0 < \alpha_S \leq 1/2$ and $c_S > 0$ such that if G is an n-vertex graph in S, then by removing at most $c_S f(n)$ edges, G can be partitioned into disjoint subgraphs G_1 and G_2 having αn and $(1-\alpha)n$ vertices respectively, where $\alpha_S \leq \alpha \leq 1-\alpha_S$.[3]

The set of removed edges is called the *cut set* of the bisection, and $f(n)$ is called the *width* of the bisection.

This definition is adequate for Lipton and Tarjan's $\sqrt{n}$-separator theorem because the class of planar graphs is closed under the subgraph relation. But there are many classes of graphs for which the same divide-and-conquer approach works, yet the class is not closed under the subgraph relation. The notion of separability can be extended by taking the closure of the original class of graphs with the subgraphs postulated by the separator theorem. Using this interpretation of separability, it is easy to show [13] that the class of trees has a 1-separator theorem. (The class of trees is not closed under the subgraph relation, although the class of *forests* of trees is.) We shall give additional separator theorems in Section 7.

[3] Throughout this paper it is assumed without loss of generality that α is chosen to permit αn to evaluate to an integral value. This assumption is preferred over the use of floor or ceiling functions because it will be useful to identify the particular values of α and because it makes the mathematical formulae more readable.

4. Areas and Aspect Ratios

The size and shape of a rectangle is uniquely determined by its *length* L and its *width* W, where we shall assume that $L \geq W > 0$. But there is another coordinate space for specifying sizes and shapes of rectangles—*area* and *aspect ratio*. Everyone is familiar with area and knows that the area can be defined as the product LW. The aspect ratio σ is defined as the quantity W/L, which by this definition, is less than or equal to one. Given the area and aspect ratio of a rectangle, its length and width are given by $L = \sqrt{A/\sigma}$ and $W = \sqrt{\sigma A}$.

Suppose a graph has a VLSI layout of area A and aspect ratio σ. It is natural to ask whether there are other layouts of the graph that have different dimensions but similar area. The following theorem shows that a long and skinny layout can be made into a square layout (aspect ratio of one) by paying only a constant factor increase in area.

> **Theorem 1:** If the bounding rectangle of a given layout has area A, then there exists topologically equivalent layout that can be enclosed in a square whose area is at most $3A$.

Proof. Let the length and width of the original layout be integers L and W. If $L < 3W$, then a square with side L satisfies the constraints of the theorem. Now suppose $L \geq 3W$. The layout can be sliced in several places and "folded" like a roadmap with the severed wires connected around the corners. Figure 4 shows a square with side $s = \lfloor \sqrt{3A} \rfloor$ in which a rectangle has been folded. This rectangle is the longest rectangle of width W that can be folded into the square, and so if we can prove that the length of this rectangle is at least L, then we will have demonstrated that the original layout can also be folded to fit in the square.

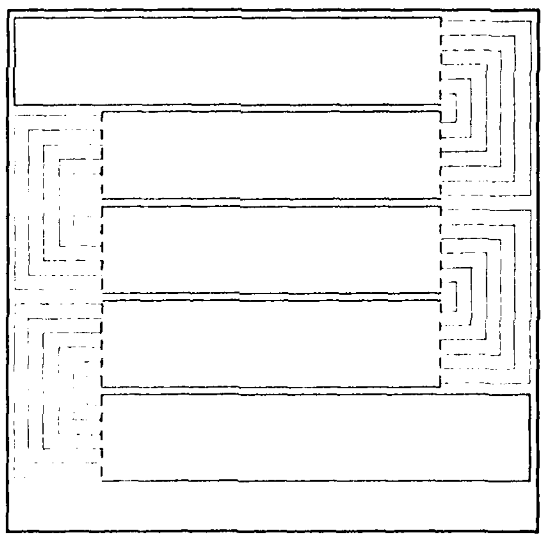

Figure 4: A layout can be "folded" to fit into a square.

Let $k = \lfloor s/W \rfloor$ be the number of pieces into which this longest rectangle of width W has been folded. The rectangle is made up of two long pieces and $k-2$ short pieces. Since $L \geq 3W$ implies $s \geq 3W$, the short pieces must be at least $s/3$ grid units long, and the long pieces must have length at least $2s/3$. Thus the total length of the folded rectangle is at least $(k-2)s/3 + 2(2s/3) = s(k+2)/3$.

Because k is the largest number of pieces of width W that can be folded into the square, it follows that

$k+1$ pieces of width W will not fit. Therefore, the length s of the side of the square must be strictly less than $W(k+1)$, which means

$$s \leq W(k+1) - 1.$$

By definition of s, the quantity $(s+1)^2$ must be strictly larger than $3A$, and hence

$$3LW \leq (s+1)^2 - 1 = s(s+2).$$

Substituting for s,

$$\begin{aligned} 3LW &\leq s(W(k+1) - 1 + 2) \\ &= s(W(k+1) + 1) \\ &\leq sW(k+2) \end{aligned}$$

since $W \geq 1$. Cancelling W from both sides and dividing by three yields $L \leq s(k+2)/3$. But the righthand side of this inequality is the value that we earlier demonstrated was less than or equal to the total length of the folded rectangle. Thus L is less than this total length, which was to be proved.[4] □

Can one "unfold" a square layout to make it arbitrarily long and skinny without paying a large increase in area? Not always, and a unit square layout provides the counterexample. If we insist that the side of the square be large, the answer is still no. For example, we showed in the introduction that an n-leaf complete binary tree can be laid out in $O(n)$ area. But in Section 9, we shall prove that the minimum dimension of that area must have order at least $\lg n$. Thus to achieve good upper bounds for layouts, it seems prudent to avoid those that have small aspect ratios.

The technique presented in Section 6 to construct area-efficient layouts recursively bisects rectangular areas. To avoid creating arbitrarily long and skinny rectangles during the recursion, it is important that the aspect ratios of the generated rectangles be bounded below by a positive constant. The next lemma sets forth conditions whereby a rectangle whose aspect ratio is so bounded can be bisected into two rectangles whose aspect ratios are similarly bounded.

> **Lemma 2:** Let R be a rectangle with area A and aspect ratio σ_R, where $\sigma_R \geq \sigma$ for some σ in the range $0 < \sigma \leq 1/2$. Suppose R is bisected parallel to its short side into two rectangles R_1 and R_2 whose areas A_1 and A_2 are ξA and $(1-\xi)A$ for some ξ in the range $\sigma \leq \xi \leq 1-\sigma$. Then the aspect ratios of the subrectangles are bounded below by σ, that is, $\sigma_{R_1} \geq \sigma$ and $\sigma_{R_2} \geq \sigma$.

Proof. Without loss of generality, we consider R_1 only. The proof may be broken into two cases. If $\xi \geq \sigma_R$, then the aspect ratio of R_1 is σ_R/ξ. This is bounded below by σ since $\sigma \leq \sigma_R$ implies that $\sigma < \sigma/\xi \leq \sigma_R/\xi$. On the other hand if $\xi < \sigma_R$, then the aspect ratio of R_1 is ξ/σ_R. But σ bounds ξ from below, and hence $\sigma < \sigma/\sigma_R \leq \xi/\sigma_R$. □

Suppose a square is divided into two rectangles so that the ratio of the area of the smaller to the larger is at worst $\sigma/(1-\sigma)$, and then the rectangles are themselves subdivided by at worst the same ratio of areas, and so forth. Lemma 2 says that if the bisection is always parallel to the short side, then no rectangle is ever generated whose aspect ratio is worse than σ. The divide-and-conquer construction in Section 6 will use this result.

[4] It should be mentioned that a worst case is achieved when a one-by-five rectangle is folded into a three-by-three square

5. A Nonlinear Recurrence

Suppose S is a class of graphs for which an $f(n)$-separator theorem has been proved. In Section 6 we shall show how to lay out any graph in S. In this section, we investigate a nonlinear recurrence equation that will be used to relate $f(n)$ to the area of the layout.

Let $A(1)$ be a positive constant, and let $A(n)$ be defined on any integer $n \geq 2$ by

$$A(n) = \max_{\alpha_S \leq \alpha \leq 1-\alpha_S} \left(A(\alpha n) + A((1-\alpha)n) + 2f(n)\sqrt{A(\alpha n) + A((1-\alpha)n)} + f^2(n)\right) \tag{1}$$
$$= \max_{\alpha_S \leq \alpha \leq 1-\alpha_S} \left(\sqrt{A(\alpha n) + A((1-\alpha)n)} + f(n)\right)^2,$$

for some $0 < \alpha_S \leq 1/2$.

Given a particular $f(n)$, there are standard methods for solving such a recurrence. We shall use a technique, however, that will enable us to solve this recurrence for broad classes of $f(n)$. We shall define a simpler function $B(n)$ which will be shown to have the property

$$A(n) \leq n B^2(n) \tag{2}$$

for all n. By providing an upper bound for $B(n)$, it will be easy to use (2) to bound $A(n)$.

We define $B(n)$ as $\sqrt{A(1)}$ for $n = 1$, and as

$$B(n) = \max_{\alpha_S \leq \alpha \leq 1-\alpha_S} \left(B(\alpha n) + f(n)/\sqrt{n}\right)$$

for $n > 1$. Property (2) holds for $n = 1$ by the definition of $B(1)$. Making the inductive assumption that it holds for values less than n,

$$\begin{aligned} A(n) &\leq \max_{\alpha_S \leq \alpha \leq 1-\alpha_S} \left(\sqrt{\alpha n B^2(\alpha n) + (1-\alpha)nB^2((1-\alpha)n)} + f(n)\right)^2 \\ &\leq \max_{\alpha_S \leq \alpha \leq 1-\alpha_S} \left(\sqrt{\alpha n B^2(\alpha n) + (1-\alpha)nB^2(\alpha n)} + f(n)\right)^2 \qquad (3) \\ &\leq \max_{\alpha_S \leq \alpha \leq 1-\alpha_S} \left(\sqrt{nB^2(\alpha n)} + f(n)\right)^2 \\ &\leq \max_{\alpha_S \leq \alpha \leq 1-\alpha_S} n\left(B(\alpha n) + f(n)/\sqrt{n}\right)^2 \\ &= nB^2(n). \end{aligned}$$

Line (3) in this proof follows from the consideration of two cases. If $B(\alpha n) \geq B((1-\alpha)n)$ for the value of α that realizes the maximum, then (3) be derived from the previous line by straightforward substitution of $B(\alpha n)$ for $B((1-\alpha)n)$. On the other hand, if $B(\alpha n) < B((1-\alpha)n)$, then substitution of $B((1-\alpha)n)$ for $B(\alpha n)$ followed by a change of variable of $1-\alpha$ for α yields the same result.

It remains to evaluate $B(n)$ which, except for the maximization, is a simple divide-and-conquer recurrence that can be solved by iteration. Thus

$$B(n) = \frac{f(n)}{\sqrt{n}} + \frac{f(\alpha_1 n)}{\sqrt{\alpha_1 n}} + \frac{f(\alpha_1\alpha_2 n)}{\sqrt{\alpha_1\alpha_2 n}} + \ldots + B(\alpha_1\alpha_2 \ldots \alpha_r n) \tag{4}$$

where $r \leq -\log_{1-\alpha_S} n$; each value $\alpha_1, \alpha_2, \ldots, \alpha_r$ is the value of α that realizes the maximum at each stage of the iteration; and the product $\alpha_1\alpha_2 \ldots \alpha_r$ equals $1/n$. Upper bounds for Equation (4) can be determined on the basis of suitable assumptions about $f(n)$. The upper bounds in Table 1 were determined by evaluating this summation according to the indicated assumptions about $f(n)$. The lower bounds for $A(n)$ were derived by defining a function $C(n)$ which is similar to $B(n)$ but which provides the bound $A(n) \geq nC^2(n)$.

Table 1: Solutions of Recurrence (1).

$f(n)$	$B(n)$	$A(n)$
$O(n^q)$, $q < 1/2$	$O(1)$	$\Theta(n)$
$\Theta(\sqrt{n}\lg^k n)$, $k \geq 0$	$O(\lg^{k+1} n)$	$O(n \lg^{2k+2} n)$
$\Omega(n^q)$, $q > 1/2$ †	$O(f(n)/\sqrt{n})$	$\Theta(f^2(n))$

†See text for an explanation of this entry.

To demonstrate the upper bound results for the third entry, it is insufficient to assume only that $f(n) = \Omega(n^q)$ for some $q > 1/2$ as the table implies. In addition the function $f(n)/\sqrt{n}$ must be well-behaved in the following sense.

Definition: A function $g(n)$ is said to satisfy *Regularity Condition C1* if there exist positive constants c_1 and β_1 such that $c_1 < 1$, $\beta_1 \leq 1/2$, and $g(\beta n) \leq c_1 g(n)$ for all sufficiently large n and all β in the range $\beta_1 \leq \beta \leq 1-\beta_1$.

Making the assumption that $f(n)/\sqrt{n}$ satisfies Condition C1 with $\beta_1 = \alpha_S$, we can now prove the third line of the table. For large n and $\alpha_S \leq \alpha_1 \leq 1-\alpha_S$, we have

$$\frac{f(\alpha_1 n)}{\sqrt{\alpha_1 n}} \leq c_1 \frac{f(n)}{\sqrt{n}},$$

and in general for each term in Equation (4)

$$\frac{f(\alpha_1\alpha_2 \ldots \alpha_k n)}{\sqrt{\alpha_1\alpha_2 \ldots \alpha_k n}} \leq c_1^k \frac{f(n)}{\sqrt{n}}.$$

Substituting these terms in Equation (4) gives the bound

$$B(n) \leq \frac{f(n)}{\sqrt{n}}(1 + c_1 + c_1^2 + \ldots) + constant,$$

which is $O(f(n)/\sqrt{n})$ since $c_1 < 1$. The constant arises from the finite number of values that are not sufficiently large according to the regularity condition.

We have just shown that the third entry in the table holds if $f(n)/\sqrt{n}$ satisfies Condition C1. What can be deduced from a weaker assumption? Suppose, for example, that we only assume that $f(n)/\sqrt{n}$ is monotonically nondecreasing, that is

$$\frac{f(\alpha n)}{\sqrt{\alpha n}} \leq \frac{f(n)}{\sqrt{n}},$$

for all $n \geq 2$ and all α in the range $\alpha_S \leq \alpha \leq 1-\alpha$. Since there are only $O(\lg n)$ terms in the summation (4), it follows that $B(n) = O((f(n) \lg n)/\sqrt{n})$ and $A(n) = O(f^2(n) \lg^2 n)$. A factor of $\lg^2 n$ in area is paid because monotonicity is a weaker constraint than Regularity Condition C1 on the well-behavedness of $f(n)/\sqrt{n}$.

The layout construction of the following section will need to assume that $A(n)$ is itself well-behaved according to a different regularity condition.

> **Definition:** A function $g(n)$ is said to satisfy *Regularity Condition C2* if there exist positive constants c_2 and β_2 such that $\beta_2 \leq 1/2$ and $g(\beta n) \geq c_2 g(n)$ for all $n \geq 2$ and for all β in the range $\beta_2 \leq \beta \leq 1-\beta_2$.

The qualification "for all $n \geq 2$" in this definition seems to be stronger than the phrase "for all sufficiently large n" which was used in the definition of Regularity Condition C1. If all the values of $g(n)$ are positive, however, the two qualifications are equivalent—although the values for the constants may be different.

Condition C2 is always satisfied by the solutions of $A(n)$ shown in the first two lines of Table 1, but not necessarily by that in the third line. To guarantee that $A(n)$ satisfies Condition C2 in this instance, it is sufficient to assume that $f(n)$ itself satisfies Condition C2 in addition to the previous assumption that $f(n)/\sqrt{n}$ satisfies C1.

The reader should be aware that most of the functions arising from a separator theorem will indeed satisfy these regularity conditions. As an example, the conditions are satisfied by all functions of the form $cn^q \lg^k n$ for constants c, q, and k such that c and q are postive. Similar regularity conditions are assumed elsewhere in the literature (e.g. [1], [3], and [4]) in order to determine the asymptotic behavior of general complexity functions.

6. Area-Efficient Layout Construction

Area-efficient layouts can be obtained through the use of the divide-and-conquer paradigm. This section presents a construction which takes a graph and divides it into two subgraphs which are recursively embedded. The two sublayouts are then sliced to expose the vertices with edges in the cut set and then those edges are routed as described in Section 2.

> **Theorem 3:** Let S be a class of graphs for which an $f(n)$-separator theorem has been proved, and let α_S and c_S be the constants postulated by the separator theorem. If $A(n)$, which is defined by $A(n) = 1/c_S^2$ for $n = 1$, and
>
> $$A(n) = \max_{\alpha_S \leq \alpha \leq 1-\alpha_S} \left(\sqrt{A(\alpha n) + A((1-\alpha)n)} + f(n)\right)^2, \tag{5}$$
>
> for $n > 1$, satisfies Regularity Condition C2 with $c_2 = \sigma_S$ and $\beta_2 = \alpha_S$, then any n-vertex graph G in S can be embedded in any rectangle whose area is at least
>
> $$A_S(n) = (4c_S^2/\sigma_S)\, A(n), \tag{6}$$
>
> and whose aspect ratio is at worst σ_S.[5]

[5] Thus the entries for $A(n)$ in Table 1 can be used to evaluate $A_S(n)$ since these two functions differ by at most a constant factor.

Proof. Let G be an n-vertex graph in S. The following recursive construction shows how to embed G in a rectangle R whose aspect ratio σ_R is at most σ_S and whose area is $A_S(n)$. Without loss of generality, view rectangle R so that the longer side which has length $\mathcal{L}(R) = \sqrt{A_S(n)/\sigma_R}$ is parallel to the horizontal axis, and so that the shorter side which has length $\mathcal{W}(R) = \sqrt{\sigma_R A_S(n)}$ is vertical.

Step 0. *Initial condition.* If $n = 1$ then the graph G is just a single vertex. Rectangle R, which has area $A_S(1)$, must contain a grid square because each dimension of R is at least two, a fact that is easily verified. Thus the theorem is true for the initial condition by simply embedding the single vertex in the grid square and returning this layout as the result of the construction.

Step 1. *Partition.* Using the $f(n)$-separator theorem, divide G into two disjoint subgraphs G_1 and G_2 which have $\alpha_G n$ and $(1-\alpha_G)n$ vertices respectively, where $\alpha_S \le \alpha_G \le 1-\alpha_S$. The number of edges in the cut set is at most $c_S f(n)$.

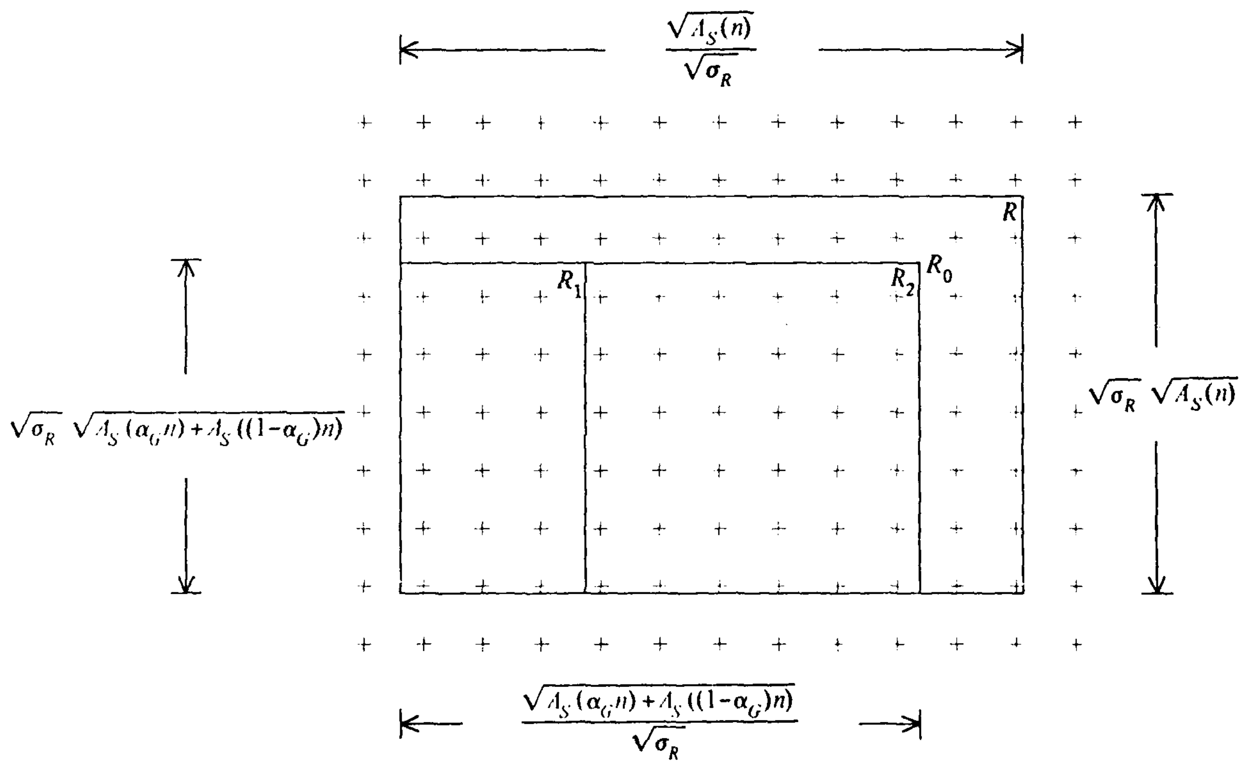

Figure 5: The relationships among rectangles in Step 2.

Step 2. *Solve the subproblems.* Remembering that rectangle R is oriented with its longer side horizontal, define R_0 to be a similar rectangle to R that has area $A_S(\alpha_G n)+A_S((1-\alpha_G)n)$ and sits in the lower left corner of R. (See Figure 5.) Apply Lemma 2 with

$$\xi = \frac{A(\alpha_G n)}{A(\alpha_G n) + A((1-\alpha_G)n)} = \frac{A_S(\alpha_G n)}{A_S(\alpha_G n) + A_S((1-\alpha_G)n)}$$

to divide R_0 into two rectangles R_1 and R_2 whose areas are $A_S(\alpha_G n)$ and $A_S((1-\alpha_G)n)$. The aspect ratios of R_1 and R_2 are bounded below by σ_S since

$$\sigma_S \le \frac{A(\alpha_G n)}{A(n)} \le \frac{A(\alpha_G n)}{A(\alpha_G n)+A((1-\alpha_G)n)} \le 1 - \frac{A((1-\alpha_G)n)}{A(\alpha_G n)+A((1-\alpha_G)n)} \le 1 - \frac{A((1-\alpha_G)n)}{A(n)} \le 1-\sigma_S,$$

which follows from the definition (5) of $A(n)$ and Regularity Condition C2. Now solve the subproblems by recursively embedding G_1 in R_1 and G_2 in R_2.

Step 3. *Marry the subproblems.* For each of the $c_S f(n)$ edges in the set of removed edges, make at most two horizontal and two vertical slices through R_0 to route the edge between its incident vertices as was shown in Figure 3. The length of this new layout is $\mathcal{L}(R_0)+2c_S f(n)$ and its width is $\mathcal{W}(R_0)+2c_S f(n)$. It remains to be shown that this layout actually fits in rectangle R, viz.

$$\mathcal{L}(R) \ge \mathcal{L}(R_0) + 2c_S f(n), \tag{7}$$

$$\mathcal{W}(R) \ge \mathcal{W}(R_0) + 2c_S f(n). \tag{8}$$

To prove these inequalities, mathematical induction can be used to give an alternative definition of $A_S(n)$ to that of Equation (6): $A_S(n) = 4/\sigma_S$ for $n = 1$, and

$$A_S(n) = \max_{\alpha_S \le \alpha \le 1-\alpha_S} \left(\sqrt{A_S(\alpha n)+A_S((1-\alpha)n)} + 2c_S f(n)/\sqrt{\sigma_S}\right)^2$$

for $n > 1$. We can now use this definition prove Inequality (7) since

$$\begin{aligned}
\mathcal{L}(R) &= \sqrt{A_S(n)/\sigma_R} \\
&\ge \sqrt{(A_S(\alpha_G n)+A_S((1-\alpha_G)n))/\sigma_R} + 2c_S f(n)/\sqrt{\sigma_S\sigma_R} \\
&\ge \mathcal{L}(R_0) + 2c_S f(n).
\end{aligned}$$

which follows from the fact that $\sigma_S\sigma_R \le 1$. The proof of Inequality (8) makes use of the fact that $\sigma_S \le \sigma_R$, whence

$$\begin{aligned}
\mathcal{W}(R) &= \sqrt{\sigma_R A_S(n)} \\
&\ge \sqrt{\sigma_R}\left(\sqrt{A_S(\alpha_G n)+A_S((1-\alpha_G)n)} + 2c_S f(n)/\sqrt{\sigma_S}\right) \\
&\ge \mathcal{W}(R_0) + 2c_S f(n)\sqrt{\sigma_R/\sigma_S} \\
&\ge \mathcal{W}(R_0) + 2c_S f(n).
\end{aligned}$$

We have shown that the layout actually fits within the bounds of rectangle R which completes the proof of Theorem 3. □

7. Corollaries of the Main Result[6]

Upper bounds on the areas of VLSI layouts for many graphs can be immediately derived as consequences Theorem 3 and Table 1. Some of these corollaries are enumerated in Table 2.

Table 2: Areas of graphs.

Class of graphs	Area of layout
Trees†	$O(n)$
Planar graphs	$O(n \lg^2 n)$
Outerplanar graphs†	$O(n)$
X-trees ($n = 2^k$)†	$O(n)$
k-dimensional meshes ($k > 2$)†	$O(n^{2-2/k})$
Graphs of genus k ($k > 0$)	$O(k^2 n \lg^2 n)$
Shuffle-exchange ($n = 2^{2^k}$)	$O(n^2/\lg n)$
Cube-connected-cycles ($n = k2^k$)†	$O(n^2/\lg^2 n)$

† These results are optimal to within a constant factor.

The separator theorems of Section 3 produce the first two results of the table. Since the class of tree graphs has a 1-separator theorem, the first line of Table 1 says that any tree or forest of trees has a layout whose area is linear in the number of vertices. Lipton and Tarjan's $\sqrt{n}$-separator theorem for planar graphs gives, according to Line 2 of Table 1, an $O(n \lg^2 n)$ area upper bound for the layout of any planar graph of n vertices.

Outerplanar graphs are triangulations of polygons, perhaps with some edges removed. The author has been able to prove a 1-separator theorem for the class of outerplanar graphs, and thus these graphs have linear area layouts. The separator theorem for trees is subsumed by this result because every tree is an outerplanar graph.

The X-tree graph [19], which is shown in Figure 6, is a complete binary tree with brother connections. One could attempt to lay out this graph by modifying the H-tree layout, but proving that the class of X-trees has a $\lg n$-separator theorem is easier. Bisect the graph with a vertical line that cuts at most $(\lg n)+1$ edges.

[6] The results reported in this section on trees and planar graphs have been discovered independently by L. G. Valiant [24]. In fact, Valiant was able to show that trees could be laid out in linear area with no crossovers. R. W. Floyd and J. D. Ullman [6] have also used similar techniques to show that any regular expression can be recognized by a linear-area circuit.

81 3 16 105

Each of the two halves can be bisected similarly, once again cutting at most $(\lg n)+1$ edges, where n is now the number of vertices in the half. Since $\lg n = O(n^q)$ for any positive q, Line 1 of Table 1 shows that any X-tree can be laid out in linear area.

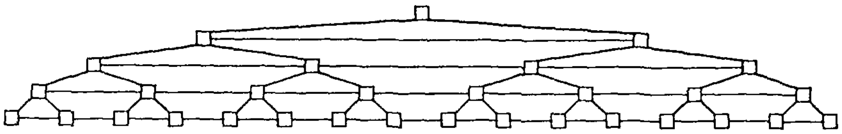

Figure 6: The X-tree on $31 = 2^5 - 1$ vertices.

A k-dimensional mesh is a graph in which each vertex is connected to its nearest neighbor in each of k dimensions. Any class of k-dimensional meshes for some constant k has an easily proved $n^{1-1/k}$-separator theorem, and thus if $k \geq 3$, an n-vertex graph in the class has an $O(n^{2-2/k})$ area layout by virtue of Line 3 of Table 1.

A graph of genus k is a graph that can be drawn with no crossovers on a sphere that has k handles attached. It has been shown [2] that there is a subset of $O(k\sqrt{n})$ vertices whose removal yields a planar graph. Applying Lipton and Tarjan's result gives a $k\sqrt{n}$-separator theorem. Line 2 of Table 1 provides an upper bound of $O(k^2 n \lg^2 n)$ for the layout area of an n-vertex graph of genus k.

In [9], Hoey and this author prove a separator theorem for the shuffle-exchange graph [20] on $n = 2^{2^k}$ vertices. Although the function in this separator theorem does not satisfy the regularity conditions of Section 5, the techniques of this paper do apply, and a $O(n^2/\lg n)$ area layout can be obtained. Recently, however, we have been able to improve this result by showing that the $O(n^2/\lg n)$ bound holds for all shuffle-exchange graphs on $n = 2^k$ vertices. This new result, however, does not the techniques in this paper.

Preparata and Vuillemin provide an $O(n^2/\lg^2 n)$ area VLSI layout for their cube-connected-cycles network [18] on $n = k2^k$ vertices. The topology of this network, which is depicted in Figure 7, can be derived from a boolean hypercube of 2^k vertices by replacing each vertex with a cycle of k vertices. This graph has a $n/\lg n$-separator theorem since removing all edges in one dimension of the original hypercube bisects the graph, removal of those in another bisects the halves, and so forth for all k dimensions. The area bound $O(n^2/\lg^2 n)$ that is given by Line 3 of Table 1 is the same as the area of the layout which is given in [18].

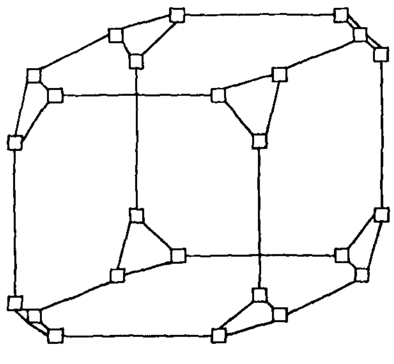

Figure 7: The cube-connected-cycles network on $24 = 3 \cdot 2^3$ vertices.

Upper bounds in Table 2 that are optimal to within a constant factor are so designated in the table. The linear upper bounds are clearly optimal because every graph requires $\Omega(n)$ area. The other lower bounds can be obtained from a result of Thompson [23]. The *minimum bisection width* of a graph is defined to be the minimum number of edges that must be cut to divide the graph into a $\lfloor n/2 \rfloor$-vertex graph and a $\lceil n/2 \rceil$-vertex graph. Thompson proves that the area of a graph has order at least the square of the minimum bisection width of the graph. This lower bound argument is surprisingly similar to an analysis of printed circuit boards given in [21].

Using another of Thompson's arguments, it can be shown that the shuffle-exchange graph and the cube-connected-cycles graph have minimum bisection widths of order at least $n/\lg n$. This arises from the fact that these networks can realize an arbitrary permutation in $O(\lg n)$ communication steps. Thus if one of these graphs is partitioned into two halves, it must be possible to swap data items between the halves in $O(\lg n)$ time. Since there are $\Omega(n)$ data items to be swapped, at least order $n/\lg n$ data cross between the halves during each time unit, and hence the minimum bisection width of these graphs is $\Omega(n/\lg n)$. The area of any VLSI layout for these graphs must therefore have order at least $n^2/\lg^2 n$. Thus the upper bound for the cube-connected-cycles graph is optimal, but there is a discrepancy in the bounds for the shuffle-exchange graph.

There is also a discrepancy in the the upper and lower bounds for planar graphs. The methods given above give only a linear area lower bound compared with the $O(n \lg^2 n)$ upper bound. The author believes it more likely that the upper bound can be improved because he knows of no planar graph that requires more than linear area, and in addition, planar graphs appear to have considerably more structure than is captured by the $\sqrt{n}$-separator theorem alone.

8. An Efficient Implementation of the Layout Algorithm

If a separator theorem can be proved for a class of graphs, Theorem 3 can be used to give an upper bound on the area of a VLSI layout for a graph in the class. If, however, a *separator algorithm* is given for the class of graphs, the steps in the proof of Theorem 3 constitute an algorithm that can construct a VLSI layout for a graph in the class. In this section, we provide an efficient implementation of this algorithm and analyze its performance.

The layout algorithm uses the separator algorithm as a subroutine, and therefore, has an execution time that depends upon the efficiencies of both this subroutine and the bookkeeping necessary for the production of a layout. The analysis here reflects this dichotomy. The total time required to lay out a graph of n vertices can be expressed as the sum of (*i*) the total time devoted to the repeated executions of the separator subroutine on the generated subgraphs plus (*ii*) the time devoted to the management of the layout representation. Later in this section, we shall present a fast bookkeeping scheme that is based on the UNION-FIND algorithm analyzed by Tarjan [22]. But first, we analyze the amount of time required by the many executions of the separator subroutine.

The layout procedure has no direct control over the efficiency of the separator subroutine. In fact, it might be the case that all the graph bisections have been previously computed so that the subroutine is deceptively fast. For the analysis here, however, we assume that the subroutine is invoked in-line, and that $s(n)$ is the time required by the separator subroutine to bisect a graph of n vertices. We can express the relationship of $S(n)$, the total amount of time required for all executions of the subroutine during the laying out of a graph of n vertices, to $s(n)$ by the recurrence $S(n) = 1$ for $n = 1$, and

$$S(n) = S(\alpha n) + S((1-\alpha)n) + s(n) \tag{9}$$

for $n > 1$, where α varies in the range $\alpha_S \le \alpha \le 1-\alpha_S$.

Bounds for $S(n)$ can be determined by the same technique used to solve Recurrence (1). Define $R(n) = S(1)$ for $n = 1$, and

$$R(n) = \max_{\alpha_S \le \alpha \le 1-\alpha_S} R(\alpha n) + s(n)/n.$$

for $n > 1$. The bound

$$S(n) \le n\,R(n),$$

which holds for the case $n = 1$, also holds for all values of n greater than one, as is shown by induction:

$$\begin{aligned} S(n) &\le \alpha n R(\alpha n) + (1-\alpha)nR((1-\alpha)n) + s(n) \\ &\le \max_{\alpha_S \le \alpha \le 1-\alpha_S} \alpha n R(\alpha n) + (1-\alpha)nR((1-\alpha)n) + s(n) \\ &\le \max_{\alpha_S \le \alpha \le 1-\alpha_S} nR(\alpha n) + s(n) \\ &\le n\,R(n). \end{aligned}$$

The results enumerated in Table 3 are derived by evaluating $R(n)$ to provide an upper bound on $S(n)$, and using a similar function to bound $S(n)$ from below. Let us look at this table in greater detail.

Table 3: Time devoted to the separator subroutine.

$s(n)$	$S(n)$
$O(n^q)$, $q < 1$	$\Theta(n)$
$\Theta(n \lg^k n)$, $k \ge 0$	$\Theta(n \lg^{k+1} n)$
$\Omega(n^q)$, $q > 1$†	$\Theta(s(n))$

†The function $s(n)/n$ must also satisfy Regularity Condition C1.

The first line is a bit of a red herring. It says that if the execution time of the separator subroutine is polynomially less than linear in the number of vertices in the graph, then the contribution to the total running time is linear. It should be apparent, however, that this precondition is rarely satisfied in practice. After all, it takes the subroutine at least linear time just to look at all of its input.

The second line of Table 3 is more usual—the subroutine requires approximately linear time. In this case, the total time required by all executions of the subroutine is only a logarithmic factor larger than the time needed by the initial invocation of the separator subroutine on the graph presented as input to the layout procedure. Tree graphs have a linear-time 1-separator algorithm that is not difficult to construct, and thus according to the table, the layout algorithm would spend a total of $O(n \lg n)$ time executing this as a subroutine when producing a layout for an n-vertex tree. It is remarkable, but Lipton and Tarjan's $\sqrt{n}$-separator algorithm for planar graphs also runs in linear time, and thus only $O(n \lg n)$ time is needed for all of its executions.

The third line of the table says that if the execution time of the separator subroutine is polynomially greater than linear, the time required by the first call, which bisects the n-vertex input graph, dominates the time for subsequent invocations. This analysis is based on the supposition that $s(n)/n$ satisfies Regularity Condition C1. When only monotonicity is assumed, the total time is $O(s(n) \lg n)$).

Now that the costs due to the $f(n)$-separator algorithm have been determined, we turn our attention to the bookkeeping required to maintain the layout representation. The implementation proposed here makes extensive use of the UNION-FIND algorithm analyzed by Tarjan [22]. This algorithm provides two instructions for the manipulation of disjoint sets. FIND(x) determines the name of the unique set containing element x, and UNION(X,Y,Z) combines the elements of sets X and Y into a new set Z. The analysis in [22] shows that the time required to execute n UNION operations intermixed with $m > n$ FIND's is $O(m\, \alpha(m,n))$ where $\alpha(m,n)$ is related to a functional inverse of Ackermann's function and grows *extremely* slowly.[7] We do not go into a description of the algorithm here—a good one can be found in [1]—but we shall use the UNION and FIND operations and the results of Tarjan's analysis.

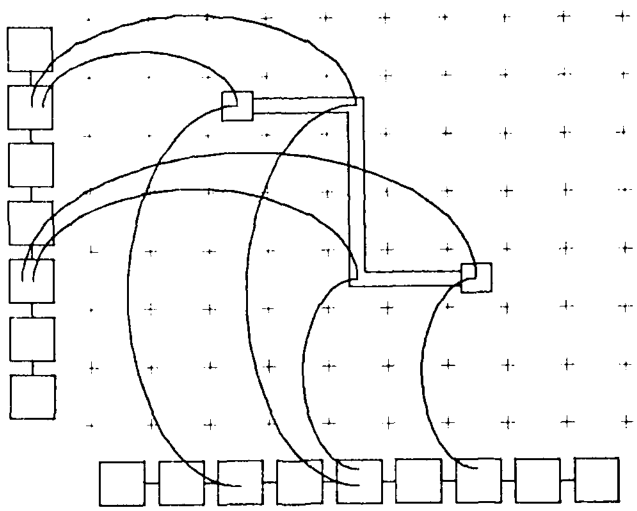

Figure 8: The representation of a layout.

The key to the performance of the layout procedure is the sparse representation of layouts depicted in Figure 8. Each important point of the layout is kept in two sets, an *x-set* which represents its x-coordinate in the layout, and a *y-set* which represents its y-coordinate. The important points in the layout are the vertices in the graph and the endpoints of the horizontal and vertical edge segments. The UNION-FIND data structure maintains the relationship between a point and its corresponding x- and y-sets. In Figure 8, this association is denoted by the curved arcs. All the x- and y-sets for a layout are kept in linked lists. The actual x-coordinate represented by a given x-set is therefore determined by its distance from the head of the list. Pointers are

[7] Tarjan comments that for all practical purposes, it is less than or equal to three.

used to maintain relationships between points. For example, an edge segment is represented by a pointer between its endpoints.

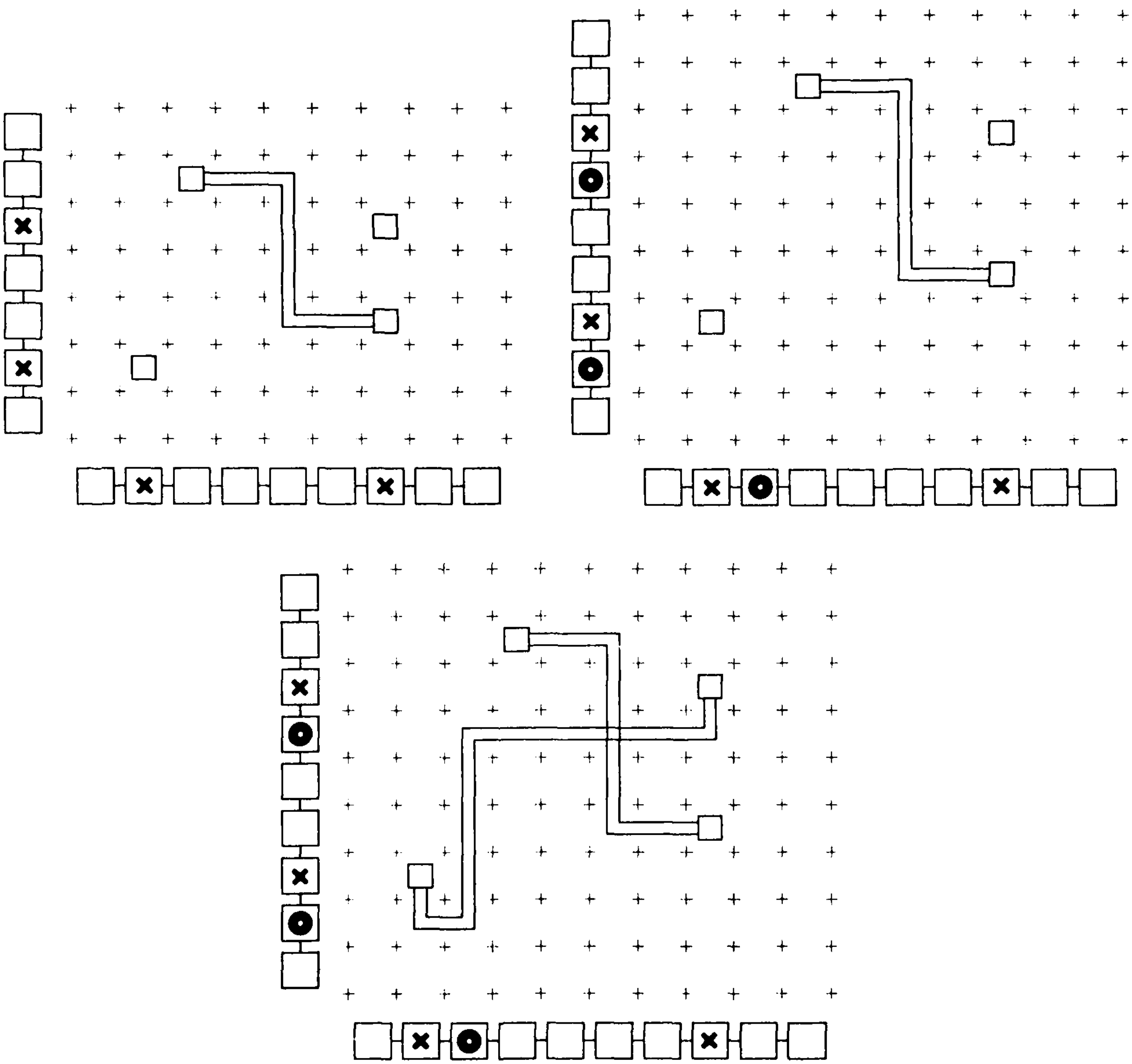

Figure 9: Routing an edge by slicing.

There are two important operations that must be performed during the layout algorithm—slicing a layout to route an edge and combining two sublayouts into a single layout. Routing a new edge between two vertices by slicing can be accomplished easily by the following procedure which is illustrated in Figure 9.

1. For each of the vertices, FIND the x-set and the y-set to which it belongs.

2. Adjacent to these x- and y-sets in the linked lists, insert new x- and y-sets, effectively adding new slices of layout. Because pointers represent the horizontal and vertical components of previously routed edges, the components are not severed and reconnected as was described in Section 2. Instead, they "stretch" automatically.

3. Add the new points for the edge to be routed to the appropriate x- and y-sets, and route the edge using pointers to represent the edge components. Each new point belongs to the x- and y-sets of the previous two steps.

Because we are considering only those classes of graphs which have bounded vertex degree, the number of edges to be routed during the entire course of execution of the layout procedure is linear in n, the number of vertices in the input graph. The routing algorithm above is called once for each edge, and hence total number of invocations is linear in n. During each invocation, a constant number of FIND's are executed, and the rest of the work takes only constant time. Thus the overall cost is the time to execute a linear number of FIND's plus another term which is linear. Since each FIND requires more than constant time, the linear number of FIND's dominates.

The cost of the FIND's cannot be determined without also knowing the number of UNION's that must be performed. The layout algorithm uses the UNION operation in the following procedure which combines two layouts into one. (Without loss of generality, assume the layouts are side-by-side in x.)

1. Append one linked list of x-sets to the other. This will produce a list of x-sets for the combined layout such that all of the x-coordinates of one sublayout lie to one side of all the x-coordinates of the other.

2. Traverse both linked lists of y-sets, and UNION corresponding y-sets to produce the linked list of y-sets for the the resultant layout. That is, the kth y-set of final layout is obtained from the UNION of the kth y-sets of the sublayouts.

The number of UNION's varies each time two layouts are combined because it is dependent upon the lengths of the linked lists that are merged. If σ_R is the aspect ratio of R, the rectangle that contains the combined layout, then the length of the linked list is $\sqrt{\sigma_R A_S(n)}$ since R is always bisected parallel to the short side. This leads to the following recurrence which describes the total number of UNION's executed by the layout algorithm: $U(n) = 0$ for $n = 1$, and

$$U(n) = U(\alpha n) + U((1-\alpha)n) + \sqrt{\sigma_R A_S(n)}$$

for $n > 1$, where α varies in the range $\alpha_S \le \alpha \le 1-\alpha_S$ and σ_R varies in the range $\sigma_S \le \sigma_R \le 1-\sigma_S$. This recurrence equation is similar to Recurrence (9) which describes time devoted to the execution of the separator subroutine. In fact, the same asymptotic results enumerated in Table 3 are valid when $\sqrt{A_S(n)}$ is substituted for $s(n)$. Notice in particular that if $A_S(n) = O(n^q)$ for some $q < 2$, then $U(n) = \Theta(n)$.

We now have a relationship between the area of the layout $A_S(n)$ and the number of UNION's $U(n)$. But $A_S(n)$ was determined, after all, by $f(n)$, the width of the separator. (Do not confuse $f(n)$ with $s(n)$, the time required to execute the separator subroutine.) Carrying this relationship through, the number of UNION's $U(n)$ can be expressed in terms of $f(n)$, and then, using the fact that there are only a linear number of FIND's, the total time required by the management of the layout representation can be determined. Table 4 enumerates these results, where $T(n)$ is the time required by the bookkeeping to lay out a graph of n vertices.

The first line of the table can be derived by observing that if $f(n) = O(n^q)$ for $q < 1$ and is monotonic if $f(n) = \Omega(\sqrt{n})$, then $A_S(n) = \Omega(n^{2q})$ and, as was noticed earlier, $U(n) = O(n)$. Because the total number of FIND's is also linear in n, the total time required for bookkeeping is $O(n\,\alpha(n,n))$.

Table 4: Time devoted to the management of the layout representation.

$f(n)$	$T(n)$
$O(n^q)$, $q < 1$†	$O(n\alpha(n,n))$
$\Theta(n)$	$O(n \lg n)$

†The function $f(n)$ must also be monotonic if $f(n) = \Omega(n^{1/2})$.

The second line of the table gives the worst-case running time for the bookkeeping which occurs when there is no better than an n-separator theorem. In this case the area given by the layout procedure is $\Theta(n^2)$, and the time to combine layouts is $O(n \lg n)$. Other bounds are readily derived for cases when the growth of $f(n)$ lies between n^q for $q < 1$ and n. For example, if $f(n) = n/\lg n$, then the time for bookkeeping is $O(n \lg \lg n)$. Thus even if the separator algorithm is only marginally good, the bookkeeping time is nearly linear.

9. Layouts with Collinear Vertices[8]

The results of previous sections can be applied to models in which different constraints are placed on the layouts. In this section, we consider layouts in which vertices are required to lie on a straight line. The results for this model can be easily generalized to other models such as that in which all vertices are constrained to lie on the (convex) perimeter of the layout. In this section, the techniques used in previous sections are employed to provide area bounds for graphs based on separator theorems for the graphs. In addition, some lower bound results are presented on the optimality of these constructions for trees and planar graphs.

Figure 10 shows how an $f(n)$-separator theorem can be used to construct a layout with collinear vertices. First, the graph is bisected by cutting at most $c_S f(n)$ edges. Then layouts are recursively constructed for the subgraphs and are placed side-by-side along the *baseline*. Vertical slices are made through the layouts, and edges are routed in the space above.

The analysis of this construction is much easier than that of Section 6. Since at most two vertical slices are made for each edge, the length of the layout along the baseline is $O(n)$. The height $H(n)$ of the layout is a constant for $n = 1$, and

$$H(n) = \max_{\alpha_S \leq \alpha \leq 1-\alpha_S} H(\alpha n) + c_S f(n)$$

for $n > 1$.

If $f(n)$ is nondecreasing, then $H(n) = O(f(n) \lg n)$ and the total area $A_S(n)$ is therefore $O(f(n)\, n \lg n)$. In particular, if $f(n) = O(\lg^k n)$, then $A_S(n) = O(n \lg^{k+1} n)$. If $f(n)$ is $\Omega(n^q)$ for some $q > 0$ and $f(n)$ satisfies Regularity Condition C1, then $H(n) = O(f(n))$ and $A_S(n) = O(n\, f(n))$.

---

[8] The the upper bounds on the areas of trees and planar graphs represent joint work with Jon L. Bentley

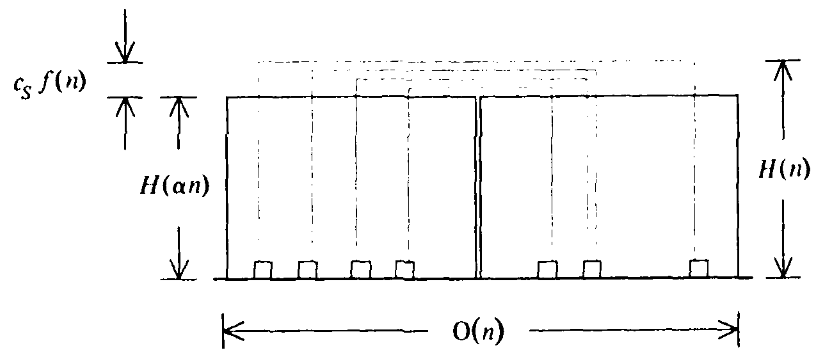

Figure 10: The construction of a layout with collinear vertices.

This means that planar graphs can be embedded on a line in $O(n\sqrt{n})$ area and trees in $O(n \lg n)$ area. We now show that these embeddings for trees and planar graphs are optimal to within a constant factor. A similar result on trees was independently discovered by Brent and Kung [5] in which they show that in any layout of a complete binary tree, the area devoted to wire must have order at least $n \lg n$. The approach here differs in that we show that the convex region containing the layout must have $\Omega(n \lg n)$ area.

> **Lemma 4:** For any complete-binary-tree layout of $n = 2^k - 1$ collinear vertices where $k \geq 0$, there exists a perpendicular to the baseline that lies between the leftmost and rightmost vertices and cuts at least $\lceil k/2 \rceil$ edges and vertices.

Proof. (Induction.) The lemma is easily satisfied for the initial cases of $n = 1$ and $n = 3$. For the general case, consider the four subtrees of size $2^{k-2}-1$. (See Figure 11.) Call the leaf that is leftmost on the baseline v, and let w be the rightmost leaf that is in a different subtree from v. Choose one of the two subtrees that contain neither v nor w. The inductive hypothesis gives us a perpendicular that cuts $\lceil (k-2)/2 \rceil$ edges or vertices in the subtree. Since v and w are in different halfplanes as determined by the perpendicular, the path between them must be cut by the perpendicular. But this path is disjoint from the subtree, which means that one more edge or vertex is cut for a total of $\lceil k/2 \rceil$. □

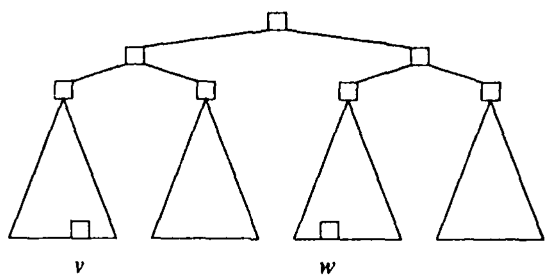

Figure 11: The construction in Lemma 4.

This lemma can be used to show that the minimum area of any convex region containing a layout for a complete binary tree must be $\Omega(n \lg n)$. The length of the layout along the baseline must be $\Omega(n)$, and as demonstrated by the previous construction, there is a point in the layout $\Omega(\lg n)$ away from the baseline. This point and the two points on the limits of the baseline determine a triangle which has $\Omega(n \lg n)$ area. Since any convex region that contains these three points must contain the triangle, so must any convex region containing the layout have $\Omega(n \lg n)$ area.

Similarly, the $O(n\sqrt{n})$ upper bound on the area for the layout of an n-vertex planar graph is tight to within a constant factor because a square mesh requires $\Omega(n\sqrt{n})$ area. This can be shown by considering that the minimum bisection width of an n-vertex square mesh is $\sqrt{n}$. Thus the perpendicular to the baseline which divides the vertices on the baseline into $\lfloor n/2 \rfloor$ and $\lceil n/2 \rceil$ vertices must cut $\sqrt{n}$ edges. The rest of the proof follows that for the complete binary tree.

The lower bound results here generalize immediately to the model in which all vertices are constrained to lie on the perimeter of a convex region. The perimeter of the region must have length $\Omega(n)$ since there are n vertices on it. The diameter of the region (the line segment which realizes the greatest distance between two points) must also be $\Omega(n)$ since it is no less than a factor of π times the length of the perimeter. Applying the previous constructions to the layout, and using the diameter of the region as a baseline yields the same lower bound results as before. In the case of the mesh, an exact bisection by a perpendicular may not be possible because some vertices may lie on the perpendicular itself. This situation can be avoided (see [23]) by putting a unit jog in the perpendicular so that it looks like a lowercase aitch without a left leg. The "perpendicular" can then be adjusted vertically to bisect the graph. For the VLSI model used in earlier sections, a similar construction shows that minimum dimension of any layout of a complete binary tree must be $\Omega(\lg n)$.

10. Perspective

Most wire-routing programs for printed circuit boards have two phases. First, the chips are placed on the printed circuit board. Then leaving the chips fixed, wires are routed one by one using heuristic search—usually a variant on the path-finding algorithm attributed to Lee [12]. Most hardware designers concede that the first of these two steps is harder. With a good placement, routing is easy; with a bad placement, routing is impossible.

Most routers for integrated circuits use much the same approach. Variations include *polycells* [15] and *gate arrays*. In the polycell approach, the components are laid down in horizontal strips and the channels between the strips are used for routing the wires. The advantage is that the channel width is not fixed. If a channel has too much congestion, extra tracks can be added easily in a manner reminiscent of slicing. The channels run both horizontally and vertically in gate arrays (also called *master slices*), but are a fixed width determined in advance. Typically, all cells are identical and are connected up with a final layer of metalization.

Recently, Johannsen [10] has introduced *bristle blocks* as a technique for laying out integrated circuits. Rather than using standard wire routing to connect cells in a design, the cells plug together. This would seem to mean that all cells must have the same width or *pitch*. Instead, however, the cells are designed with places to stretch so that a cell with smaller pitch can be adjusted to plug into a wider cell with no routing necessary.

The idea of using divide-and-conquer to help with the general wire-routing problem is not new. As far back as 1969, Günther [8] gave a heuristic procedure for arranging machines in a workshop given the frequency of travel between machines. This algorithm, which applies as much to circuit placement as to machine placement, partitions the transportation graph and places the subgraphs in subrectangles of the original area. Günther's technique for partitioning is highly heuristic, and he comments that it is the critical step. Another heuristic for graph partitioning is given by Kernighan and Lin [11]. Among the applications they mention is that of partitioning chips among printed circuit boards so as to minimize the connections between boards. There is an algorithmic solution to the partitioning problem, however. It is based on the fact that the graphs of interconnections which arise in practice are almost planar. By replacing each crossover in some drawing of the graph with an artificial vertex that performs the crossover, Lipton and Tarjan's separator algorithm for planar graphs can be applied.

It is unlikely that a fast general partitioning algorithm will be found because the problem of finding the minimum bisection width of a graph is NP-complete [7]. In other words, graphs are hard to partition. This unfortunate situation brings up the question, *"Can the divide-and-conquer approach used in this paper, which performs placement and routing simultaneously compete with or enhance those techniques already in use?"*

A difficulty with applying the techniques of this paper concerns constant factors in the areas of layouts. The model in Section 2 assumes that each vertex fits into a square of the grid, and furthermore, that the sizes of vertices and edges are comparable. For many practical applications, the vertices are somewhat larger than the edges. This means that the grid size is substantially larger than the edge width, and thus each slice through the layout wastes a large constant factor. A solution to this problem is to design the cells represented by vertices with places where they can be sliced, and then use the largest unsliceable portion of a cell as the granularity of the grid. This technique complements the bristle blocks approach because places where a cell can stretch are frequently places where it can be sliced.

There is another solution, however, which does not require the cells to be sliceable, and yet does allow the granularity of the grid to be the width of a wire. The limitation is that sizes and shapes of vertices must not vary widely. Each vertex is placed in a rectangle whose area is four times the area of the vertex. The layout algorithm is allowed to slice this rectangle, but slicing is allowed only in one direction. In the other direction the space between or next to the layouts is used as a channel for routing. When a slice is made through a vertex, the vertex is not sliced, but instead the edge simply crosses over. When the algorithm terminates, each edge that crosses over a vertex is routed around the vertex in the unused area provided by the rectangle. The author is currently working on another approach based on weighted separator theorems where at each stage of the recursion, all edges that are to be routed at a higher level are brought to the periphery of the current layout.

Where vertices are large, unsliceable, and of widely varying sizes, the problem becomes one of two-dimensional bin-packing with constraints. This formulation seems the least tractable. It may be, however, that as with bin-packing, simple heuristics can be found that give reasonable solutions for commonly occuring instances.

Acknowledgements

I would like to thank H. T. Kung for his copious comments on several drafts of this paper, Jon L. Bentley for steering me in the right directions, Bernd Bruegge for his help in translating [8], Earl Mounts for his aid in library searches, and especially James B. Saxe for the idea of introducing $B(n)$ to prove the bounds on Equation (1) and for suggestions that led to a simpler proof of Theorem 1.

References

1. Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, Massachusetts, 1974.
2. Michael O. Albertson and Joan P. Hutchinson, "On the independence ratio of a graph," *Journal of Graph Theory*, Vol. 2, 1978, pp. 1-8.
3. Jon Louis Bentley, Dorthea Haken, and James B. Saxe, "A general method for solving divide-and-conquer recurrences," Technical report CMU-CS-78-154, Department of Computer Science, Carnegie-Mellon University, December 1978.
4. R. P. Brent and H. T. Kung, "Fast algorithms for manipulating formal power series," *Journal of the Association for Computing Machinery*, Vol. 25, October 1978, pp. 581-595.
5. R. P. Brent and H. T. Kung, "On the area of binary tree layouts," Technical report TR-CS-79-07, The Australian National University, Department of Computer Science, July 1979.
6. Robert W. Floyd and Jeffrey D. Ullman, "The compilation of regular expressions into integrated circuits," February 1980, draft.
7. M. R. Garey, D. S. Johnson, and L. Stockmeyer, "Some simplified polynomial complete problems," *6th Annual Symposium on Theory of Computing*, ACM, April 1974, pp. 47-63.
8. Th. Günther, "Die räumliche anordnung von einheiten mit wechselbeziehungen," *Elektronische Datenverarbeitung*, May 1969, pp. 209-212.
9. Dan Hoey and Charles E. Leiserson, "A layout for the shuffle-exchange network," *1980 International Conference on Parallel Processing*, August 1980.
10. Dave Johannsen, "Bristle blocks: a silicon compiler," *Caltech Conference on Very Large Scale Integration*, January 1979, pp. 303-310.
11. B. W. Kernighan and S. Lin, "An effective heuristic procedure for partitioning graphs," *Bell Systems Technical Journal*, Vol. 49, February 1970, pp. 291-308.
12. C. Y. Lee, "An algorithm for path connection and its applications," *IRE Transactions on Electronic Computers*, Vol. EC-10, No. 3, September 1961, pp. 346-365.
13. P. M. Lewis, R. E. Stearns, and J. Hartmanis, "Memory bounds for recognition of context-free and context-sensitive languages," *IEEE Symposium on Switching Circuit Theory and Logical Design*, IEEE, 1965.
14. Richard J. Lipton and Robert E. Tarjan, "A separator theorem for planar graphs," *A Conference on Theoretical Computer Science*, University of Waterloo, August 1977.
15. Roland L. Mattison, "A high quality, low cost router for MOS/LSI," *Proceedings of the ACM-IEEE Design Automation Workshop*, Dallas, Texas, June 1972, pp. 94-103.
16. Carver A. Mead and Lynn A. Conway, *Introduction to VLSI Systems*, Addison-Wesley, Reading, Massachusetts, 1980.
17. Carver Mead and Martin Rem, "Cost and performance of VLSI computing structures," *IEEE Journal of Solid State Circuits*, Vol. SC-14, No. 2, April 1979, pp. 455-462.
18. Franco P. Preparata and Jean Vuillemin, "The cube-connected-cycles: a versatile network for parallel computation," Technical report 356, Institut de Recherche d'Informatique et d'Automatique, June 1979.
19. C. H. Séquin, A. M. Despain, and D. A. Patterson, "Communication in X-tree, a modular multiprocessor system," *ACM 78 Proceedings*, ACM, 1978.
20. Harold S. Stone, "Parallel processing with the perfect shuffle," *IEEE Transactions on Computers*, Vol. C-20, No. 2, February 1971, pp. 153-161.

21. Ivan E. Sutherland and Donald Oestreicher, "How big should a printed circuit board be?," *IEEE Transactions on Computers*, Vol. C-22, May 1973, pp. 537-542.

22. Robert Endre Tarjan, "Efficiency of a good but not linear set union algorithm," *Journal of the Association for Computing Machinery*, Vol. 25, No. 2, 1975, pp. 215-225.

23. C. D. Thompson, *A Complexity Theory for VLSI*, Ph.D. dissertation, Department of Computer Science, Carnegie-Mellon University, 1980.

24. L. G. Valiant, "Universality considerations in VLSI circuits," December 1979, draft (to appear in *IEEE Transactions on Computers*).

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER CMU-CS-80-138	2. GOVT ACCESSION NO. AD-A096 367	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) AREA-EFFICIENT GRAPH LAYOUTS (FOR VLSI)		5. TYPE OF REPORT & PERIOD COVERED Interim
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) CHARLES E. LEISERSON		8. CONTRACT OR GRANT NUMBER(s) N00014-76-C-0370
9. PERFORMING ORGANIZATION NAME AND ADDRESS Carnegie-Mellon University Computer Science Department Pittsburgh, PA 15213		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Office of Naval Research Arlington, VA 22217		12. REPORT DATE August 1979 (revised 8/80)
		13. NUMBER OF PAGES 26
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

DD FORM 1473, 1 JAN 73 EDITION OF 1 NOV 65 IS OBSOLETE

S/N 0102-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)