LABORATORY FOR
COMPUTER SCIENCE

MASSACHUSETTS
INSTITUTE OF
TECHNOLOGY

MIT/LCS/TR-305

A FRAMEWORK FOR SOLVING VLSI GRAPH LAYOUT PROBLEMS

Sandeep N. Bhatt
Frank T. Leighton

This research was supported in part by the Defense Advanced Research Projects Agency of the Department of Defense and was monitored by the Office of Naval Research under Contract No. N00014-80C-0662.

545 TECHNOLOGY SQUARE, CAMBRIDGE, MASSACHUSETTS 02139

This blank page was inserted to preserve pagination.

A Framework For Solving VLSI Graph Layout Problems

Sandeep N. Bhatt [1]

Frank Thomson Leighton [1,2]

[1]Laboratory for Computer Science
and
[2]Department of Mathematics
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

Abstract: This paper introduces a new divide-and-conquer framework for VLSI graph layout. Universally close upper and lower bounds are obtained for important cost functions such as layout area and propagation delay. The framework is also effectively used to design regular and configurable layouts, to assemble large networks of processors using restructurable chips, and to configure networks around faulty processors. It is also shown how good graph partitioning heuristics may be used to develop a provably good layout strategy.

Key Words: bifurcator, bisection width, capacitive delay, configurable layout, crossing number, decomposition tree, graph layout, layout area, mesh of trees, propagation delay, separator theorem, tree of meshes, Very Large Scale Integration (VLSI), wafer-scale integration, wire length.

This research was supported in part by DARPA contract N00014-80-C-0622. Tom Leighton was also supported in part by a Bantrell Fellowship and Air Force contract AFOSR-82-0326.

1. Introduction

The tremendous engineering advances made in Very Large Scale Integration (VLSI) fabrication technology have stimulated considerable theoretical interest in VLSI circuit layout problems. Most of this effort has centered on minimizing the layout area of a circuit on a chip. This is due, in part, to the fact that layouts which consume large amounts of chip area are more expensive to fabricate, less reliable and harder to test than layouts which consume smaller amounts of chip area.

Other layout-related issues that have been studied include: minimizing propagation delay (either by decreasing wire lengths or by increasing transistor sizes), minimizing the number of wire crossings in a layout, producing regular layouts for gate-arrays, designing chips that can later be configured to realize a large number of circuits, configuring networks around defective cells on a wafer, and assembling large systems of processors from copies of a single basic chip which has few external pin connections.

Most theoretical techniques devised thus far are based on the divide-and-conquer paradigm and require the use of a separator theorem to recursively partition a given circuit. Although separator-based techniques work well for some graphs, they perform very poorly for others.

In this paper we propose an alternative framework for solving VLSI graph layout problems. Like previous approaches, the new framework is based on the divide-and-conquer paradigm. Instead of using a separator theorem to recursively partition a graph, the new framework requires the use of a *bifurcator*. The difference between bifurcators and separators will, of course, be explained in the paper, but the two primary advantages of bifurcators over separators may be stated here. First, unlike separators, bifurcators may be efficiently computed using either a good graph partitioning heuristic, or from a layout with small area. Second, bifurcators can be used to produce layouts that are efficient in a variety of respects, not layout area alone.

For example, using the notion of bifurcators, an area-efficient layout can be transformed into a layout which is both area-efficient and also has small propagation delay. The same result can also be achieved if, instead of an area-efficient layout, we use an efficient graph bisection heuristic. Separator theorems are inherently weaker than bifurcators for such purposes, and no other approach is known to enjoy the versatility of bifurcators.

This paper is based on, and unifies the work contained in three extended abstracts by Bhatt and Leiserson [3, 4] and Leighton [21]. Although the results are self-contained, some familiarity with recent results in VLSI layout theory would be helpful in reading this paper. A fairly comprehensive list of recent research papers is included in the references. In particular, Ullman [43] provides a good introduction to issues in VLSI layout theory.

The paper is divided into nine sections. In Section 2, we review the layout model and the separator-based approach to VLSI layout. In Section 3, we formally state eight VLSI layout problems and briefly review the progress made on each problem. The combinatorial lemmas proved in Section 4 provide the basis of the new framework described in Section 5. In Section 6, we describe how the framework can be used to efficiently solve the eight layout problems described in Section 3. Section 7 shows how a good graph bisection heuristic can be used to produce a provably good layout strategy. In Section 8, we prove that the upper bounds for area,

crossing number and minimax edge length found in Section 6 are existentially optimal. The paper concludes with some remarks and open questions in Section 9.

2. Background

Thomspon [41, 42] provided the first formal model for VLSI circuit layout. The model is simply stated and captures the important aspects of layout problems in a realistic manner. A brief description of the model is included in Section 2.1. In addition, Thompson also proved some elementary upper and lower bounds on the area required to lay out an arbitrary graph, which are discussed in Section 2.2. More general bounds were obtained later by Leiserson [26, 27] and Valiant [45], who independently developed a divide-and-conquer layout strategy based on separator theorems. Section 2.3 summarizes their results and highlights a major deficiency of any layout scheme based on separator theorems.

2.1. The Layout Model

In order to cast VLSI layout problems within a mathematical framework, Thompson [41, 42] developed a formal model for VLSI graph layout. The model is based on, and is consistent with, the VLSI design rules established by Mead and Conway [31]. It is also similar to the widely used Manhattan wiring model. In the *Thomspon grid model*, a layout for a graph is characterized as an embedding within a *two-dimensional grid*. A *two-dimensional grid* is a collection of horizontal and vertical *tracks* spaced apart at unit intervals. A layout for a graph G is specified by an *embedding* which assigns nodes of G to points in the grid where horizontal and vertical tracks intersect, together with an (incidence-preserving) assignment of the edges of G to *paths* in the grid. The paths of the layout are restricted to follow along grid tracks and are not allowed to overlap for any distance (although a vertical path segment may cross a horizontal path segment). In addition, the paths may not cross nodes to which they are not adjacent. For obvious reasons, we restrict our attention to graphs in which no node has degree greater than four. As an example, Figure 1 shows a layout for the complete graph on four nodes.

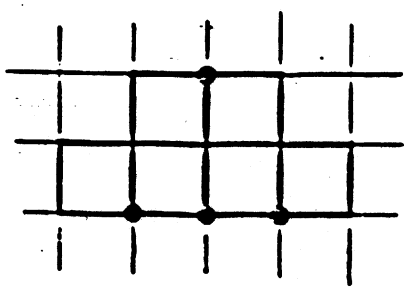

Figure 1. *A layout with area 15.*

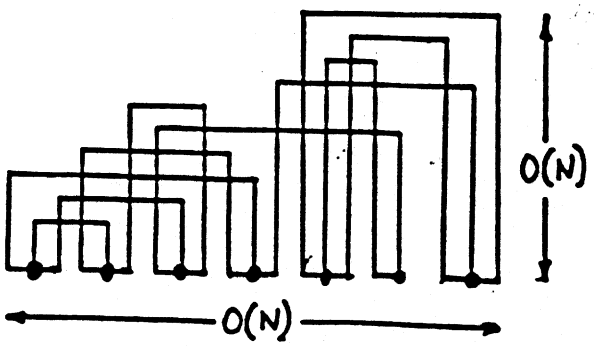

Figure 2. *Every N-node graph can be laid out in $O(N^2)$ area.*

Remark. The results in this paper easily extend to variants of the Thomspon grid model. For example, graphs with bounded valence greater than four may be laid out by mapping each node to a region of the grid, instead of a single grid point. The results are also applicable to networks with large processors. Techniques for dealing with large processors are described more fully in the discussion of Problem 5 in Sections 3 and 6.

2.2. Elementary Bounds on Layout Area

Although there are a variety of important engineering considerations in choosing one layout for a graph over other possible layouts, the best understood, and perhaps the most desirable cost measure to minimize is *layout area.* The area of a layout is most naturally defined as the area of the "bounding-box" around the layout, and equals the product of the number of vertical tracks and the number of horizontal tracks that contain a node or wire segment of the graph. For example, the layout of Figure 1 has area 15. This is not the minimum possible; there is another layout with area 9.

How much area does an N-node graph require? Clearly, the area cannot be less than the number N of nodes. On the other hand, by embedding nodes at equally spaced intervals along a line, and using a distinct horizontal track for each edge (as shown in Figure 2), it is clear that the area required for an N-node graph is no greater than $O(N^2)$. These bounds are independent of the structure of the graph and hold for all N-node graphs. In general, however, the minimum area needed to lay out a graph depends on the graph.

Thompson [41, 42] identified *bisection width* as an important property of graphs that affects minimum layout area. The bisection width of a graph is the minimum number of edges which must be removed from the graph in order to disconnect it into two equal-size pieces. (Two graphs are said to be of *equal size* if the difference in the numbers of nodes is no more than one.) Thompson showed that, up to a constant factor, the layout area can be no less than the square of the bisection width. Therefore, if the bisection width for a graph is known, a lower bound

on area can be easily computed. By showing that certain computationally powerful graphs such as the shuffle-exchange graph have large bisection width, Thompson showed that these graphs require large area. In fact, Thompson extended this observation to obtain area-time tradeoffs for computing certain functions.

Leighton [19, 20] identified *crossing number* as another general property that affects layout area. The crossing number of a graph is defined as the minimum number of edge crossings in any drawing of the graph in the plane. It is easy to see that the crossing number of a graph is a lower bound on layout area. Using more sophisticated arguments for special graphs, Leighton also directly obtained lower bounds on *total wire length* (the sum of the lengths of the wires in a layout), which of course is a lower bound on layout area. These techniques are heavily dependent on the recursive structure of the special graphs and will be generalized in Section 8.

2.3. Layouts Based on Separator Theorems

Leiserson [26, 27] and Valiant [45] investigated general properties that provide effective upper bounds on layout area. They independently developed a divide-and-conquer strategy for graph layout and showed, for example, that every N-node tree can be laid out in $O(N)$ area and that every N-node planar graph can be laid out in $O(N \log^2 N)$ area. Their technique is based on the notion of separator theorems for graphs.

> *Definition:* A class of graphs which is closed under the subgraph relation is said to have an *$f(x)$-separator theorem* if there exist constants a and b where $0 < a \leq 1/2$ and $b > 0$ such that every N-node graph in the class can be partitioned (by the removal of at most $bf(N)$ edges of the graph) into disjoint subgraphs having $a'N$ and $(1 - a')N$ nodes where $a \leq a' \leq 1 - a$.

Given a class of graphs for which a separator theorem is known (e.g., trees have a 1-separator theorem [28] and planar graphs have a $\sqrt{x}$-separator theorem [29]), it is possible to construct a layout for any N-node graph in the class by using a simple divide-and-conquer approach. For example, Leiserson [26, 27] proved the following upper bounds on layout area.

x^α-separator theorem	*Layout Area*
$\alpha < 1/2$	$O(N)$
$\alpha = 1/2$	$O(N \log^2 N)$
$\alpha > 1/2$	$O(N^{2\alpha})$

Remark. The complete recursive decomposition of a graph must be provided as input before layouts achieving the desired area bounds can be constructed by the procedure. There is no polynomial time algorithm known that achieves the area bounds if the decomposition is not provided. This severely limits the applicability of separator-based layout strategies to classes of graphs (such as trees or planar graphs) for which actual decompositions are known.

How good are the preceding area bounds? Thompson [41, 42] and Leighton [19, 20] showed

that none of the bounds can be improved. More precisely, they showed that within each class there is a graph for which the bound is optimal. But this does not mean that the bounds are optimal for every graph within a class. In fact, while the bounds are *existentially* optimal, they are not *universally* optimal. For example, an N-node square grid requires area N, but since the minimum separator theorem for the class of square grids is $\sqrt{x}$, the best bound obtainable by separator-based layouts is $O(N \log^2 N)$, which is off by a factor of $O(\log^2 N)$ from the optimal. Of course, since N-node graphs require area at least N, the bounds for graphs with x^α-separator theorems, where $\alpha < 1/2$, are asymptotically universally optimal.

For graphs with larger separator theorems, the discrepancy between the minimum layout area and that given in the table can be much worse. Consider, for example, the N-node graph S_N which consists of $N/\log N$ disjoint $\log N$-node expander graphs. We define an m-node *expander graph* to be a graph for which any subset of k nodes is linked by $\Theta(\min(k, m-k))$ edges to the $m-k$ nodes outside the subset. The bisection width of such a graph is $\Omega(m)$, and hence the minimum separator theorem is $\Theta(x)$. The existence of trivalent graphs that satisfy this defintion has been known for a long time [12, 15, 44]. In fact, almost all trivalent graphs satisfy this definition. We caution the reader that the term "expander graph" has two definitions in the literature. The other definition is sufficient for our purposes and probably more standard but requires graphs with higher node degrees. Since each $\log N$-node expander graph can be trivially laid out in $O(\log^2 N)$ area, the layout area of S_N is no greater than $O(N \log N)$. However, Leighton [21] showed that the minimum separator theorem for the class of graphs S_N exceeds $\Omega(x/\log^2 x)$, so that the area bound from the table above is $O(N^2/\log^4 N)$, which is much worse than the optimal bound of $O(N \log N)$.

Remark. The careful reader will notice (as did the referee) that any class of graphs closed under the subgraph relation and containing S_N, must also contain expander graphs. Hence, the minimum separator for the class is $\Theta(x)$. In order to get around such technicalities with the definition, the concept of a separator is often just applied to a single graph and the subgraphs produced by its recursive decomposition. Using the less restrictive (but more useful) definition, it is possible to show that S_N has an $O(N/\log N)$-separator. The $\log N$-node expander graphs are split in the upper levels of the decomposition and never appear intact as subgraphs in the lower levels of the decomposition. Leighton proved that even using the most liberal definition, the minimum separator for S_N is at least $\Omega(N/\log^2 N)$. Any bound on layout area for S_N based on the minimum separator can be no less than $\Omega(N^2/\log^4 N)$.

Thus, while the divide-and-conquer strategy based on separator theorems gives existentially optimal bounds, the bounds can be unacceptably poor in a universal sense. It was the discovery of such large discrepancies that led to the search for an alternative framework for VLSI layout. Within the new framework presented in Section 5 we shall see how these large discrepancies are overcome.

3. Eight VLSI Graph Layout Problems

As mentioned earlier, there are many important considerations in choosing one layout over a

multitude of other possible layouts. The problems in this section are motivated by some of the basic engineering concerns. Although this list is not meant to be exhaustive, it covers most of the theoretical issues studied recently. Many of the problems are known to be NP-Complete, so the solutions we later obtain will, of course, not be optimal. Rather, the major emphasis of this paper is the development of a general framework for handling layout problems efficiently and in a uniform manner. Within the framework, solutions to some problems are close to optimal. For other problems, good heuristics are developed and/or general bounds are obtained.

Problem 1. *Given a graph G, produce an area-efficient layout for G.*

As mentioned before, minimizing area is a critical concern in VLSI circuit layout. In addition to the work on area-efficient layouts described in the previous section, Dolev, Leighton, and Trickey [9] have shown that determining the minimum layout area of a forest of trees is NP-Complete.

Problem 2. *Given a graph G, produce an area-efficient layout for G with minimax edge length.*

Besides area, speed is another critical factor in chip performance. Signals do not propagate instantaneously across wires, and the longer the wire, the longer the propagation delay. In pipelined or systolic systems, the effect of propagation delays is even more dramatic. The maximum delay determines the clockperiod, and hence the throughput, of the system. To maximize throughput we need to minimize the maximum delay. In short, we must produce layouts so that the longest edge is as short as possible. The minimum, over all layouts, of the length of the longest edge is called the *minimax* edge length.

Paterson, Ruzzo and Snyder [34] studied the problem of minimizing edge lengths for complete binary trees. They showed that the minimax edge length of an N-node complete binary tree is $\Theta(\sqrt{N}/\lg N)$. Adopting a different strategy based on separator theorems, Bhatt and Leiserson [3] subsequently extended the upper bound portion of the result to arbitrary trees, and to all graphs with small (i.e., x^{α}, $\alpha < 1/2$) separator theorems. Bhatt and Cosmadakis [2] showed that computing the minimax edge length of a tree is NP-Complete.

Problem 3. *Given a graph, produce an area-efficient layout in which each wire has bounded delay in the capacitive model.*

Although it is certainly true that propagation delay across a wire depends on the length of the wire, there has been little consensus on how fast propagation delay grows as a function of wire length. Thompson [41, 42] assumes propagation delay to be constant, independent of wire length. This might seem unreasonable given the ultimate speed-of-light limitation which indicates that the delay increases linearly with length. The spped-of-light limitation, however, greatly exaggerates the importance of wire delay in determining the speed of circuits. Mead and Conway [31] take into account some of the electrical characteristics of interconnections on MOS integrated circuits, and emphasize the role of wire *capacitance* in determining propagation delay. Recent analysis by Bilardi, Pracchi, and Preparata [5] strongly supports the belief that capacitive effects play the predominant role in determining the speed of MOS circuits.

In a capacitive model, each wire is assumed to present a purely *capacitive load* to the transistor that drives a signal across thewire. This load is proportional to the length of the wire plus the area of the transistor that receives the signal. The delay is proportional to this load divided by the area of the driving transistor. By increasing the size of the driving transistor it is therefore possible to bound the propagation delay, independent of the length of the wire. A second well-known technique for reducing delay across a long wire is to "ramp" the wire with a geometrically increasing series of inverters [31]. The number of intermediate drivers, and hence the delay, is logarithmic in the length of the wire, but an attractive feature is that this process can be carried out without the need to resize the original transistors in the circuit.

Of course, increasing the size of one transistor or introducing new transistors might force some wires to be stretched to avoid the enlarged transistor area. In other words, decreasing the delay across one wire might force an increase in delay over other wires. Leiserson [24] and Mehlhorn [32] independently posed the question of whether or not the transistors in a layout could be resized so that every wire in the layout has constant propagation delay. Ramachandran [36] investigated the problem of introducing intermediate drivers along long wires to decrease delays, but under the constraint that the topology of the layout remain unchanged. With the restriction that wires can not be rerouted, she showed that logarithmic delay can be achieved, but at the expense of squaring the layout area in the worst case. We allow the layout topology to be changed, and obtain significantly better results.

Problem 4. *Given a graph G, produce a layout for G with few wire crossings.*

An undesirable feature of layouts is the presence of a large number of wire crossings. When two wires cross, they must be on different layers. For faster operation, and less power dissipation, it is advantageous to maximize the total amount of wiring on a layer of low resistance, e.g. the metal layer, while minimizing the wiring on a layer of high resistance, e.g. the polysilicon layer. The net wiring on one layer may be reduced by laying wires on that layer only just before and after two wires cross. If the number of wire crossings is small, the number of contact-cuts which connect wire segments on different layers is small so that the area of the layout is not blown up by the contact cuts which occupy large area. In addition, long wires that are crossed by many other wires are susceptible to cross-talk when all the crossing wires simultaneously carry the same signal.

The *crossing number* of a graph is defined to be the minimum number of wire crossings in any drawing of the graph on the plane. Leighton [19, 20] proved upper and lower bounds on crossing numbers and then used the results to find bounds on layout area. Garey and Johnson [14] showed that determining the crossing number of bipartite graphs is NP-Complete.

Problem 5. *Given a graph, produce an area-efficient regular layout for the graph.*

Some design methodologies, most notably gate-arrays, require that processors be located at fixed positions on a chip. In gate-arrays the processors are placed in a grid pattern with uniform spacing between processors adjacent along every row and column. Such layouts are said to be *regular*. An important advantage of this design restriction is its flexibility: even if the size of every processor is increased, the wiring between processors remains unaffected and the total

area remains proportional to the *sum* of the wire area (as computed with unit-size processors) and the processor area. This is because only the $\sqrt{N}$ rows and columns containing the N unit-size processors need to be expanded to accomodate the non-unit-size processors. In non-regular layouts, *every* row and column might have to be expanded since there might be a node in every row and in every column. Increasing the linear dimension of the processors by a factor of s could result in an $\Theta(s^2)$ increase in layout area.

Previous divide-and-conquer layout strategies do not produce regular layouts. Hence, they are not useful in laying out circuits with non-unit-size processors. A good strategy for producing regular layouts would solve the nagging problem of how to cope with variable-size processors.

Problem 6. *Design area-efficient chips that can be configured to realize a large number of graphs.*

Because it is expensive to make one chip but cheap to make many copies, manufacturers of custom chips have been encouraged to make *configurable* designs such as gate-arrays, ROM's and PLA's. In such designs, the entire chip is prefabricated except for one layer. The customer then specifies a configuration for the chip, and the final layer of metalization connects up the circuitry in that particular way. Hence, most of the design and fabrication costs can be factored over many custom chips. Similarly, the fast emerging laser-restructuring technology [35] provides another economical way to customize chips after fabrication is complete. Laser restructuring allows connections between wires to be made or broken after the chip has been fabricated. In either case, it is desirable to design layouts that can be configured from one of a few basic patterns.

Problem 7. *On a wafer which has arbitrarily distributed defective cells, realize a given graph on the good cells.*

In any fabrication process, it is expected that some of the processing cells will be defective. In a two-dimensional array of cells on a wafer in which defective cells are arbitrarily distributed, it may still be possible to use the wafer by configuring wires around the defective cells. This may, for example, be performed by laser restructuring techniques [35]. Given this ability to isolate defective cells, it is important to consider how a graph may be realized on the remaining good cells. This problem has received considerable attention recently [17, 22, 38]. The problem is similar to the general graph layout problem in the Thompson model but with the important restriction that nodes of the circuit can only be mapped to a restricted set of nodes in the grid.

Problem 8. *Given a graph G, assemble G using the minimum number of copies of a single chip having few external pin connections.*

A number of very large networks have been proposed in recent years for implementing priority queues [25], for searching [1], for direct execution of applicative programming languages [30], and for recognizing regular expresions [11]. Some of these networks are too large to fit on a single chip. For example, the tree-structured network of [30] is envisioned to contain as many as one million processing elements. Clearly, such networks must be partitioned over many interconnected chips, so that each chip realizes a small portion of the network.

The technology for packaging chips severely limits the number of external pin connections

on a chip. While chips with over a million components are forseeable in the near future, no one predicts a chip with over two hundred external pin connections. This poses a pressing problem in assembling large networks of processors.

Even if a network could be partitioned so that each portion has only a few external connections, it would be economically infeasible to design each chip individually. For instance, it would be prohibitively expensive to design one thousand different chips, each containing a thousand processing elements, to assemble a network of one million processors. For this reason, it is necessary to assemble large systems using copies of a few configurable or restructurable chips. One solution to the problem of assembling large tree structures using copies of a single, area-efficient, restructurable chip with few external pin connections was given by Bhatt and Leiserson [4].

Within the new framework, efficient solutions are provided for each of these problems. In fact, a single layout simultaneously solves many of these problems efficiently. The framework provides a two-step strategy for solving these problems. First, the graph to be laid out is embedded within a very special network called the *tree of meshes*. For the tree of meshes it is possible to solve all these problems efficiently. In the second step therefore, a good layout for the tree of meshes also solves these problems for the embedded graph.

4. Combinatorial Lemmas

This section contains three combinatorial lemmas which provide the foundation for the framework presented in the next section.

Lemma 1. *Consider any two-ended string of n colored pearls of k different colors, and let n_i be the number of pearls which are color i for $1 \leq i \leq k$. For any integer $r \geq 2$, the pearls can be partitioned into two sets by cutting the string in no more than $9r^k$ places such that the total number of pearls in each set is $\lfloor n/2 \rfloor$ or $\lceil n/2 \rceil$, the number of pearls of color 1 in each set is $\lfloor n_1/2 \rfloor$ or $\lceil n_1/2 \rceil$, and such that the number of pearls of color $i > 1$ in each set lies between $\lceil (\frac{1}{2} - \frac{1}{2r})n_i \rceil$ and $\lfloor (\frac{1}{2} + \frac{1}{2r})n_i \rfloor$.*

Proof. Let i be a number between 1 and k and let $T(i)$ denote the number of cuts necessary to divide the set of all pearls into two sets that satisfy the constraints of the theorem for colors $1, 2, \ldots, i$. Other than requiring that the total number of pearls be split in half by the cuts, we have made no constraints on the distribution of pearls with colors greater than i. We wish to find a good bound on $T(i)$ in the worst case, i.e., over all choices of $n, k \geq i$, and all possible colorings. In what follows, we will show that $T(1) = 2$ and that

$$T(i) \leq rT(i-1) + 4r + 7$$

for $i > 1$. As a consequence, we can solve the recurrence to conclude that $T(i) \leq 9r^i - 15$ for $r \geq 2$. Thus for $i = k$, at most $9r^k$ cuts are required, as claimed.

For $i = 1$, it is easy to show that two cuts are sufficient. Consider a "window" of size $\lfloor n/2 \rfloor$ positioned at the left end of the string. Without loss of generality, assume that the window

covers less than $\lfloor n_1/2 \rfloor$ of the pearls colored 1. Move the window to the right, one pearl at a time until the window covers $\lfloor n_1/2 \rfloor$ pearls of color 1. Since the right half of the string contains more than one-half of all pearls of color 1, there must, by continuity, exist a placement when the window covers exactly one-half of all pearls of color 1. By cutting the string at the endpoints of the window, the portion of the string under the window will contain half of the total number of pearls and half of the pearls colored 1. Hence $T(1) = 2$, as claimed.

For a given $i > 1$, break the string into r segments S_j, $1 \leq j \leq r$, (making $r - 1$ cuts) so that each segment contains at least $\lfloor n_i/r \rfloor$ pearls of color i. Next split each S_j into two subsets S_{j0} and S_{j1} (making a total of $rT(i-1)$ cuts) so that each split satisfies the theorem locally for colors 1, 2,..., $i - 1$.

Without loss of generality, assume that S_{j0} contains no fewer pearls of color i than S_{j1}. At this stage, we divide the set C of all pearls into two subsets C_1 and C_2 as follows. Initially, let $C_1 = \bigcup S_{j0}$. If C_1 contains more than $\lfloor (\frac{1}{2} + \frac{1}{2r}) n_i \rfloor$ pearls of color i, remove S_{10} from C_1 and add S_{11}. Repeat this procedure, successively switching S_{20} with S_{21}, S_{30} with S_{31}, and so on until the first time C_1 has at most $\lfloor (\frac{1}{2} + \frac{1}{2r}) n_i \rfloor$ pearls of color i. Such a stage must occur since the number of pearls of color i in C_1 will eventually fall below $\lceil n_i/2 \rceil$ if C_1 and C_2 are completely interchanged. The number of pearls of color i in C_1 after the final switch cannot be less than $\lceil (\frac{1}{2} - \frac{1}{2r}) n_i \rceil - 2$ since every S_j contains no more than $\lceil n_i/r \rceil$ pearls of color i. If the number of pearls of color i in C_1 is $\lceil (\frac{1}{2} - \frac{1}{2r}) n_i \rceil - 1$ or $\lceil (\frac{1}{2} - \frac{1}{2r}) n_i \rceil - 2$, then move either one or two pearls of color i from C_2 to C_1, making no more than four cuts.

We also have to ensure that the total set of pearls and the pearls of the first $i - 1$ colors are divided as required. The pearls with colors between 2 and $i - 1$ are divided correctly because they were divided correctly at the recursive step. The counts of pearls of color 1 in C_1 and C_2 may differ in size by r, however. To balance the number of pearls with color 1 in each set, we need only remove up to $\lfloor r/2 \rfloor$ pearls colored 1 from the excess set (making at most r cuts) and put them in the deficient set. To balance the difference in the overall sizes of the sets (which now might be as large as $2r + 4$), we need only extract up to $r + 2$ pearls from the larger set (making no more than $2r + 4$ cuts) and put them in the smaller set. Of course, these pearls must be chosen carefully so that each set retains the required minimum number of pearls of each color. Since pearls are extracted only from the larger set, it is clear that this requirement may be easily satisfied.

The total number of cuts made by the procedure is $rT(i-1) + 4r + 7$, as claimed. ∎

Using an elegant topological argument, Goldberg and West [16] recently proved that k cuts suffice to divide the pearls of each color exactly in half. In contrast to Lemma 1, this is a dramatic reduction in the number of cuts. We state their result in Lemma 2, although we cannot include the proof here. We will use the stronger result in the paper since it facilitates the proofs and results in far smaller constants. It is very important to note, however, that *all of our layout results may be proved with the weaker Lemma 1.* (In fact, we have done so using $r = 3$, but will not go through the details in this paper.) Since the Goldberg-West result has not yet appeared, we have included Lemma 1 both for completeness and so that our results will not depend on as-yet unpublished work. Both results are implementable in polynomial time when the number of colors is fixed, as is the case throughout this paper.

Lemma 2. *Consider any two-ended string of n pearls, n_i of which are colored i, $1 \leq i \leq k$. By cutting the string in k places it is possible to divide the pearls into two sets so that each set has a total of $\lfloor n/2 \rfloor$ or $\lceil n/2 \rceil$ pearls, and $\lfloor n_i/2 \rfloor$ or $\lceil n_i/2 \rceil$ pearls of color i for all i, $1 \leq i \leq k$.*

The following lemma recasts Lemma 2 in terms of complete binary trees. This form is particularly useful since the recursive decomposition of a graph may be viewed as a tree. In the following we define the *height* of a tree to be the length of the longest path from the root to a leaf. The *height* of a forest is defined to be the maximum height of a tree in the forest. Finally, the *level* of a node in the forest is defined to be the height of the forest minus the length of the longest path from the node to a leaf. (Note that the top level is level zero.)

Lemma 3. *Consider a forest of complete binary trees whose n leaves are colored arbitrarily with k colors. Let n_i be the number of leaves colored i for $1 \leq i \leq k$. By removing no more than k nodes (as well as all incident edges) from each internal level of the forest, it is possible to produce a new forest of complete binary trees, some subset of which contains $\lfloor n/2 \rfloor$ or $\lceil n/2 \rceil$ leaves, and $\lfloor n_i/2 \rfloor$ or $\lceil n_i/2 \rceil$ nodes of color i for each i, $1 \leq i \leq k$.*

Proof. Draw the trees in the canonical manner and place them side-by-side, in any order, so that the leaves of all trees are placed along a line. By applying Lemma 2 to the induced left-to-right ordering on the leaves of the forest, it is possible to break the ordering in no more than k places such that the union of the leaves contained in every other segment contains the desired total number of leaves and the desired number of leaves of each color.

For each break, remove the nodes (and incident edges) which are simultaneously ancestors of the leaf immediately to the left of the break and the leaf immediately to the right of the break. It is easily seen that at most one node is removed from each internal level of the forest for each break. Therefore, no more than k total nodes are removed from each internal level. In addition, the removal of the common ancestors of the leaves neighboring a break divides the associated tree into two or more complete binary trees, at least one on each side of the break. Thus the removal of all such nodes produces a forest of complete binary trees, subsets of which correspond precisely to the sets of leaves between pairs of adjacent break points. Thus the union of the subsets of trees corresponding to every other segment of leaves contains the desired number of leaves of each color. ∎

Figure 3 illustrates the proof of the preceeding lemma with a simple example. Initially, the forest consists of four complete binary trees with seven leaves colored 1, four colored 2, and four colored 3. Figure 3a shows a leveled drawing of the forest along with three breaks (denoted by dashed vertical lines) in the line of leaves. The union of leaves in the first and third intervals contains three leaves colored 1, two of color 2, and two of color 3. In Figure 3b the internal nodes to be removed are marked X. Figure 3c shows the new forest produced by the removal of the marked internal nodes.

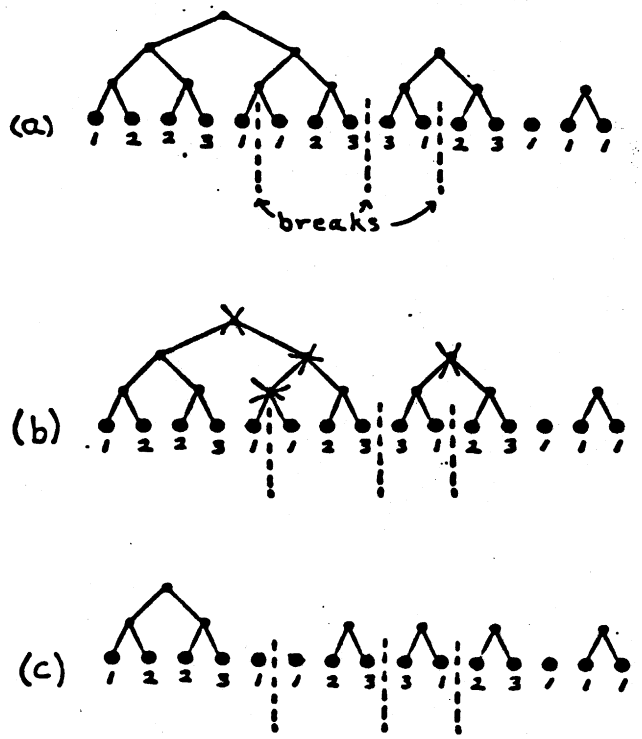

Figure 3. *An illustration of the procedure described in Lemma 9.*

5. The New Framework

In this section, we describe the new framework for solving VLSI graph layout problems. We start by defining the notions of *decomposition trees* and *bifurcators* for graphs. Using the combinatorial lemmas from Section 4, we devise procedures for *balancing* decomposition trees and bifurcators. In Section 5.3, balanced decomposition trees are used to embed graphs within the *tree of meshes*. Section 5.4 provides efficient layouts for the tree of meshes. Taken together, the embedding of a graph in the tree of meshes and the layout for the tree of meshes induce a layout for the original graph.

5.1. Decomposition Trees and Bifurcators

The recursive decomposition of a graph into smaller and smaller subgraphs may be viewed as a decomposition tree. In particular, we say that a graph G has an $(F_0, F_1, \ldots, F_r)$-*decomposition tree* if G can be decomposed into two subgraphs G_0 and G_1 by removing no more than F_0 edges from G, and, in turn, both G_0 and G_1 can be decomposed into smaller subgraphs by removing no more than F_1 edges from each, and so on until each subgraph is either empty or an isolated node. Figure 4 illustrates this recursive decomposition.

As one might expect, the decomposition of a graph by separator theorems may be viewed as a decomposition tree. It follows by definition that if a class of graphs has an $f(x)$-separator theorem, then there are constants α and β such that each graph in the class has a decomposition tree of the form $(\beta f(N), \beta f(\alpha N), \beta f(\alpha^2 N), \ldots, \beta f(1))$. The converse is not necessarily true. Subgraphs generated at each step of a decomposition by a separator theorem are constrained to

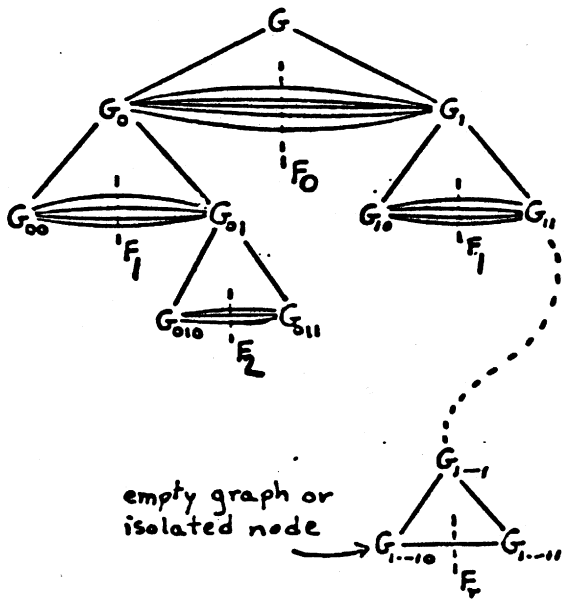

Figure 4. *An $(F_0, F_1, \ldots, F_r)$-decomposition tree.*

be proportional in size, whereas decomposition trees need not satisfy this constraint. Of course, if the decomposition tree has precisely $\log N$ levels, then subgraphs at each level must be equal in size.

We shall be particularly interested in a special class of decomposition trees, namely *bifurcators*, that is distinct from the class of separators.

Definition. An N-node graph has an α-bifurcator of size F (more simply, an (F, α)-bifurcator) if it has an $(F, F/\alpha, F/\alpha^2, \ldots, 1)$-decomposition tree.

Of particular interest is the class of $\sqrt{2}$-bifurcators. By the definition, we know that an N-node graph has a $\sqrt{2}$-bifurcator of size F if and only if it has an $(F, F/\sqrt{2}, F/2, \ldots, 1)$-decomposition tree. The depth of this tree is no greater than $2 \log F$. In order to completely decompose an N-node graph into individual nodes, the height of any decomposition tree cannot be less than the $\log N$. Thus, F must always be at least $\sqrt{N}$. On the other hand, F is always less than $2N$ since every N-node graph with maximum node degree four has at most $2N$ edges.

If a class of graphs has an x^α-separator theorem, where $\alpha \leq 1/2$, and the corresponding decomposition is *balanced* in that every graph is always decomposed into equal-size subgraphs, then it is straightforward to show that every N-node graph in the class has a $\sqrt{2}$-bifurcator of size $O(\sqrt{N})$. Similarly, if a class of graphs has a balanced separator theorem of size x^α with $\alpha > 1/2$, then every N-node graph in the class has a $\sqrt{2}$-bifurcator of size $O(N^\alpha)$.

The converse is not true even if we consider only bifurcators whose corresponding decomposition trees are balanced so that every graph is decomposed into equal-size subgraphs. For example, the N-node graph S_N defined in Section 2.3 has a balanced $\sqrt{2}$-bifurcator of size $O(\sqrt{N \log N})$ but the smallest separator for this class of graphs is $\Omega(x/\log^2 x)$.

When translated into bounds on layout area, this seemingly minor difference between bifurcators and separators is greatly magnified. *Graphs with small layout area always have small $\sqrt{2}$-bifurcators, but do not always have small separators.* This is formalized in the following lemma. Later on we will prove the converse: graphs with small $\sqrt{2}$-bifurcators always have small layout area.

Lemma 4. *If a graph G can be laid out in area A, then G has a $(\sqrt{A}, \sqrt{2})$-bifurcator.*

Proof. Consider a vertical cut of length $\sqrt{A}$ through the center of the layout. Next, cut each of the sublayouts horizontally through the center. Continuing this sequence of alternating vertical and horizontal cuts, it is easy to see that at the ith step no more than $\sqrt{A}/2^{\lfloor i/2 \rfloor}$ edges are cut from each subgraph. This sequence of cuts yields a $(\sqrt{A}, \sqrt{2})$-bifurcator for G. ∎

5.1.1 Special Cases

Many graphs have decomposition trees in which the number of cuts decreases very slowly as we go lower down the tree. In such cases the number of cuts at higher levels of the tree may be very small. On the other hand, in decomposition trees corresponding to bifurcators, the number of cuts permitted decreases smoothly as we go down the tree. It is conceivable then, that the bifurcator permits far more cuts at higher levels than are necessary. For example, N-node binary trees have decomposition trees of height $O(\log N)$ in which no more than 1 cut is required at every level. Since the minimum bifurcator is at least $\sqrt{N}$, the decomposition tree corresponding to the bifurcator allows far more cuts at the top levels than needed.

Similarly, some graphs have decomposition trees in which many cuts are required at the top levels, but this number decreases very quickly as we go down the decomposition tree. In such cases, the minimum bifurcator is large so that decomposition trees corresponding to the bifurcator do not underestimate the number of cuts required at the top level. However, they do greatly overestimate the number of cuts at lower levels.

It is useful to separate such extreme cases from a general discussion. Of course, general upper bounds are valid for graphs with extreme decompositions, but they may overestimate the true bound. A particularly important reason for separating these classes is that many computationally useful graphs such as binary trees fall into the first category while cube-connected-cycles and multidimensional meshes fall into the second category.

An N-node graph is defined to have a *type A $\sqrt{2}$-bifurcator* if it has an $(O(\sqrt{N}), \sqrt{2})$-bifurcator such that no more than $O((N/2^i)^\alpha)$ cuts, $\alpha < 1/2$, are required for each partition at the ith level of the associated decomposition tree. Observe that at the higher levels of the tree, $i << \log N$, the number of cuts is far less than the $O(\sqrt{N}/2^{i/2})$ cuts allowed by the usual bifurcator.

Similarly, an N-node graph is defined to have a *type B $\sqrt{2}$-bifurcator* if it has an $(O(N^\alpha), \sqrt{2})$-bifurcator, $\alpha > 1/2$, such that only $O((N/2^i)^\alpha)$ edges are cut in any partition at the ith level. Observe that for the lower levels of the tree, $i >> 1$, this quantity is far smaller than the $O(N^\alpha/2^{i/2})$ cuts allowed by the usual bifurcator.

For simplicity, we will prove results only for general $\sqrt{2}$-bifurcators in this paper. However, whenever there is a significant difference, results for the special cases are stated separately. The

proofs for these special cases are easily worked out, and closely follow the proofs for the general cases. We leave such details to the interested reader.

5.2 Balanced Decomposition Trees

Of particular interest to the layout results reported in this paper are decomposition trees where at each step of the decomposition, the two subgraphs are nearly equal in size. This section considers such balanced decompositions and gives an effective procedure for transforming an arbitrary decomposition tree into one that is balanced.

Formally, a decomposition tree for a graph G is *balanced* if each subgraph G_w in the tree is the father of two subgraphs G_{w0} and G_{w1} such that the number of nodes in the subgraphs differ by at most 1. In addition, we say that a decomposition tree is *fully balanced* if it is balanced, and if for every subgraph G_w in the tree, the set of edges connecting $G - G_w$ to G_w is divided into two subsets of nearly equal size by the partition of G_w into G_{w0} and G_{w1}. (Here we allow the number of edge connections in the two subgraphs to differ by a small constant, say 5. For the purposes of simplicity, however, we shall often ignore such small differences and assume that the nodes and connections are split evenly between the two subgraphs.)

Somewhat surprisingly, any decomposition tree may be transformed into a fully balanced one at little or no cost. We prove this in the following theorem which generalizes earlier results in [4, 19, 20, 21].

Theorem 5. *Let G be any N-node graph with an $(F_0, F_1, \ldots, F_r)$-decomposition tree T. Then G has a fully balanced $(F'_0, F'_1, \ldots, F'_{\log N})$-decomposition tree, such that for $0 \leq i \leq \log N$,*

$$F'_i = 6 \sum_{s=i}^{r} F_s.$$

Proof. Let Γ be a forest of complete binary trees consisting initially of the decomposition tree T. Color the leaves of T with two colors according to whether or not the subgraph of G associated with the leaf is empty. Apply Lemma 3 ($k = 2$) to Γ, removing the indicated nodes and edges of T. Each node of T corresponds naturally to a set of edges of G, namely the edges whose removal splits the associated subgraph in two. Removing a node of T corresponds to removing this cutset of edges from G. Since no more than 2 nodes are removed from each level of Γ, the number of edges removed from G in applying Lemma 3 does not exceed $2\sum_{s=0}^{r} F_s$, which is less than F'_0.

Further note that G is divided into two disjoint subgraphs of nearly-equal-size by the removal of these edges. Each subgraph, in turn, corresponds in a natural way to a subforest of complete binary trees in Γ. Consider one such subgraph G_0 and color the leaves of the associated forest of complete binary trees Γ_0 using six colors as follows:

> If the leaf corresponds to an empty subgraph, color the leaf with color 1. Otherwise, if the single node corresponding to the leaf is incident to exactly j edges of G removed earlier, $0 \leq j \leq 4$, then color the leaf with color $j + 2$.

By applying Lemma 3 ($k = 6$) to Γ_0, it is clear that G_0 can be decomposed into two disjoint

subgraphs G_{00} and G_{01} of nearly-equal-size such that the number of edges from $G - G_0$ to G_{00} is nearly-equal to the number of edges from $G - G_0$ to G_{01}. Since at most 6 nodes were removed from each level of Γ_0 and since Γ_0 does not contain the root of T, we can conclude that no more than $6\sum_{s=1}^{r} F_s = F'_1$ edges were removed from G_0.

By applying the above argument recursively, the desired fully-balanced decomposition tree is easily obtained. The only point to observe is that with each application of Lemma 3, the biggest tree in any forest corresponding to a subgraph decreases in height by at least one. This is because the total number of leaves in each forest is cut in half at each step. A total of $\log N + 1$ levels are sufficient for the decomposition since the number of nodes in each subgraph is also split in half at each step. ∎

Theorem 6. *Every graph with a $\sqrt{2}$-bifurcator of size F has a fully balanced $\sqrt{2}$-bifurcator of size $6(2+\sqrt{2})F$.*

Proof. The result follows immediately from the preceeding theorem, with the observation that $\sum_{i\geq 0} 2^{-i/2} \leq 2+\sqrt{2}$. ∎

Remark. The procedure described in Theorems 5 and 6 can be implemented in polynomial time.

5.3 Embeddings in the Tree of Meshes

Leighton [19, 20] introduced the tree of meshes as an example of a planar graph that cannot be laid out in linear area. He also showed that every N-node planar graph can be embedded in an $O(N \log N)$-node tree of meshes. In this section, we define the tree of meshes and describe a general strategy for embedding a graph in the tree of meshes.

The *tree of meshes* is formed by replacing each node of a complete binary tree with a mesh and each edge by several edges which connect meshes at consecutive levels. More precisely, the root of the complete binary tree is replaced by an $n \times n$ mesh (it is assumed that n is a power of 2), the nodes at the second level are replaced by $n \times n/2$ meshes, those at the third level by

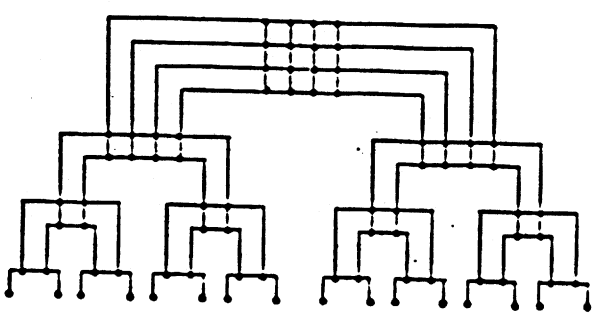

Figure 5. *The 4×4 tree of meshes T_4.*

$n/2 \times n/2$ meshes, and so on until the leaves of the tree are replaced by 1×1 meshes. As shown in Figure 5, each edge of the tree is replaced with edges connecting nodes on one side of the higher-level mesh to the top row of the mesh at the lower level. The resulting graph is called the $n \times n$ tree of meshes T_n. It is not difficult to see that T_n, has $N = 2n^2 \log n + n^2$ nodes.

For some applications, we need to consider only the top levels of the tree of meshes. We call the subgraph consisting of levels $0, 1, \ldots, p$ of T_n a *truncated tree of meshes* $T_{n,p}$. Note that $p \leq 2 \log N$.

Theorem 7. *There is a constant c such that every N-node graph G with an $(F, \sqrt{2})$-bifurcator can be embedded in $T_{cF, 2 \log \frac{N}{F}}$. Moreover, the embedding is regular in the sense that F^2/N nodes of G are embedded in a regular fashion each of the N^2/F^2 bottom-level meshes of $T_{cF, 2 \log \frac{N}{F}}$.*

Proof. We first use Theorem 6 to construct a fully-balanced $\sqrt{2}$-bifurcator of size $6(2+\sqrt{2})F$ for G. We then use the internal meshes of $T_{cF, 2 \log \frac{N}{F}}$ to route the edges that were removed in the upper $2 \log \frac{N}{F}$ levels of the fully balanced decomposition tree for G. The subgraphs in the $(2 \log \frac{N}{F})$th level of the decomposition tree (each of which has $\lfloor F^2/N \rfloor$ or $\lceil F^2/N \rceil$ nodes) are then embedded in the meshes on the bottom level of the truncated tree of meshes.

The internal meshes are used in the same manner that complete crossbar switches are used in switching networks. For example, in Figure 6 six wires enter the mesh through the top, of which four exit from the left side and two from the right. In addition, four wires enter and exit from the sides. No matter what the ordering of the wires, they can easily be routed through the mesh as shown. In general, if the number of wires routed through a mesh does not exceed any side-length of the mesh, a routing may always be found. Similarly, a graph with M nodes can always be embedded in a $4M \times 4M$ mesh with nodes placed in a regular fashion.

Consider only the top $2 \log \frac{N}{F} + 1$ levels of a fully balanced decomposition tree for G. Each of the subgraphs at level $2 \log \frac{N}{F}$ of the decomposition tree has $N(1/2)^{2 \log \frac{N}{F}} = F^2/N$ nodes. (For simplicity we shall assume that F^2/N is an integer.) Furthermore, if E_i is the maximum number of edges between $G - G_i$ and G_i, where G_i is a subgraph in the decomposition tree at level i, then it is easy to see that $E_0 = 0$ and by Theorem 6, that

$$E_i \leq \frac{1}{2} E_{i-1} + 6(2+\sqrt{2}) \frac{F}{2^{(i-1)/2}}$$

for $1 \leq i \leq 2 \log \frac{N}{F}$. Solving the above recurrence, we obtain:

$$E_i \leq 6(2+\sqrt{2}) \frac{F}{2^{(i-1)/2}} \sum_{s \geq 0} (\sqrt{2}/2)^s,$$

and thus

$$E_i \leq 6(2+\sqrt{2})^2 \frac{F}{2^{(i-1)/2}}.$$

We now embed G in $T_{cF, 2 \log \frac{N}{F}}$. First, embed each of the $(2 \log \frac{N}{F})$-level subgraphs of the decomposition tree in the bottom level meshes. This can be done if the side of each mesh at level

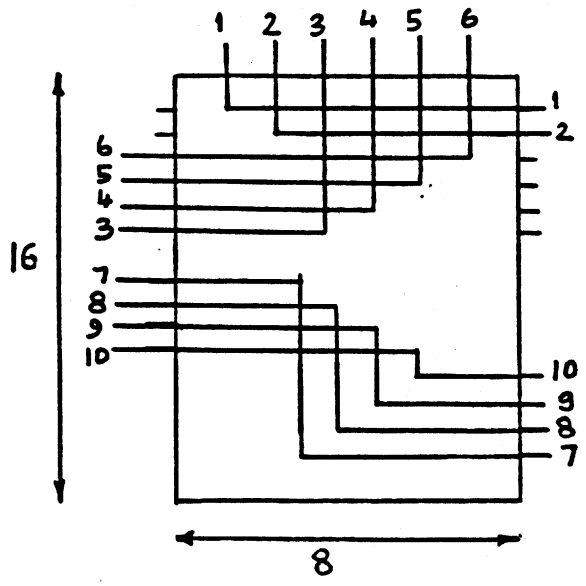

Figure 6. *Using a mesh in the tree of meshes as a crossbar switch.*

$2\log\frac{N}{F}$ exceeds $4F^2/N$. This is true provided

$$cF/\sqrt{2}^{2\log\frac{N}{F}} \geq 4F^2/N.$$

For $c \geq 4$, this inequality is easily satisfied.

Next embed the additional edges through the upper-level meshes in the natural way. No more than $2E_{i+1}$ edges pass through any ith level mesh. Thus the routing can be performed if the smaller side of the ith level meshes exceeds $2E_{i+1}$. In other words, we must have:

$$cF/2^{\lceil i/2 \rceil} \geq 12(2+\sqrt{2})^2 F/2^{i/2}.$$

A simple calculation shows that the inequality is satisfied for sufficiently large c. ∎

Remark. Throughout the paper, we express bounds using the term $\log\frac{N}{F}$. For all practical purposes, F is much smaller than N and this term is greater than one. Should the value of F be larger, however, we shall still define $\log\frac{N}{F}$ to be at least one. Similar interpretations are assumed for $\log\log\frac{N}{F}$ and for $\log\log\log\frac{N}{F}$. The conventions avoid the annoying (and trivial) cases when F is very large without complicating the analysis further.

In the preceding embedding, all the nodes of G were mapped to meshes at the bottom level of the truncated tree of meshes. Thus, edges between nodes in different meshes might have to be routed through as many as $4\log\frac{N}{F}$ meshes. Such long edges are undesirable for a variety of reasons. It is natural to ask whether an embedding can be found in which each edge can be routed through fewer intermediate meshes. This is answered in the following theorem.

Theorem 8. *There are constants c and k such that every N-node graph G with an $(F, \sqrt{2})$-bifurcator can be embedded in $T_{cF, 2\log\frac{N}{F}}$ and such that no edge is routed through more than k intermediate meshes.*

Proof. We adopt a slight variant of the strategy used in Theorems 5-7. The balancing and embedding are done simultaneously and in the same manner as before, except at levels 0, $k, 2k, 3k, \ldots$ (where k is a constant specified later). At these levels, we embed the nodes that are incident to edges previously cut, and we cut the previously uncut edges incident to these nodes. Of course, this could triple the number of cut edges every k levels but if k is sufficiently large, this happens infrequently and is not harmful. At all other levels the procedure is the same as before, using 6 colors and Lemma 3 to partition the decomposition tree. The process terminates after $2\log\frac{N}{F}$ levels.

As before, the embedding is accomplished by using meshes as switching boxes for routing edges. We must ensure that the number of edges routed through any mesh does not exceed the side lengths of the mesh. The calculation is the same as before except that the number of cut edges is tripled at every kth level. Thus the recurrence for E_i is

$$E_i \leq \frac{1}{2}(3^{1/k})E_{i-1} + 6(2+\sqrt{2})\frac{F}{2^{(i-1)/2}}.$$

Here, we have (without loss of generality) increased number of cut edges by a factor of 3 initially and by a factor of $3^{1/k}$ at each level instead of increasing the number of cuts by a factor of 3 at every kth level. Solving the recurrence, we find

$$E_i \leq 18(2+\sqrt{2})\frac{F}{2^{(i-1)/2}}\sum_{s\geq 0}\left(\frac{\sqrt{2}}{2}3^{1/k}\right)^s.$$

For $k \geq 4$, the sum converges to a constant. The remaining analysis is the same as in Theorems 5-7, except that the constants are larger. ∎

Remark. It is worthwhile to point out here that Theorems 7 and 8 could also have been proved using Lemma 1 as instead of Lemma 2. The nodes of G would still be balanced in the decomposition tree but the cut edges could only be split 1/3 - 2/3 at each decomposition. While this increases the value of the sum, it still converges to a constant. (This is because for sufficiently large k, $\frac{2\sqrt{2}}{3}3^{1/k} < 1$.) Hence, k and c would be larger but the statements of the theorems remain the same.

5.4 Layouts for the Tree of Meshes

Thus far we have considered only the problem of embedding graphs in the tree of meshes. How do we lay out the tree of meshes efficiently? Clearly, any layout for the tree of meshes also gives a layout for every graph that can be embedded within the tree of meshes. In this section we develop two different layouts for the tree of meshes.

The first layout is a straightforward modification of the "H-tree" layout for complete binary trees [31]. The modified layout is obtained by expanding each node of the complete binary tree

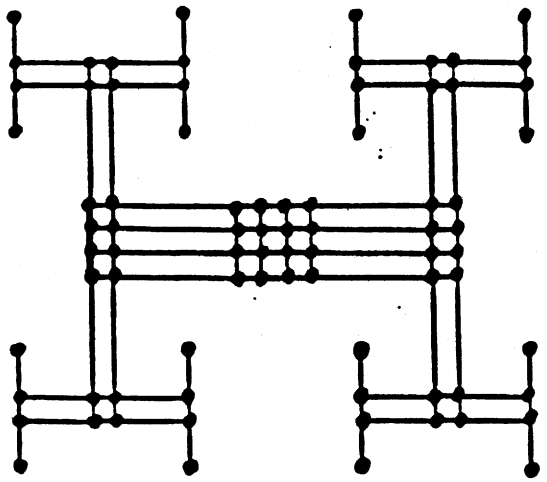

Figure 7. *The H-layout of the tree of meshes.*

into a mesh of the appropriate size. Figure 7 shows this layout. It is easy to see that if $S(F)$ denotes the side of the layout for T_F, then $S(1) = 1$, and

$$S(F) \leq 2S(F/2) + O(F),$$

which gives $S(F) = O(F \log F)$. This means that the area of the layout for T_F is bounded by $O(F^2 \log^2 F)$. As shown in [19, 20], this bound is optimal.

For truncated trees of meshes, such as considered in Theorems 7 and 8, a similar result holds.

Theorem 9. *The truncated tree of meshes $T_{F,2\log\frac{N}{F}}$ has a layout of area $O(F^2 \log^2 \frac{N}{F})$.*

Proof. The obvious restriction of the H-layout to the top levels suffices. ∎

Although the mesh edges in the layout shown in Figure 7 have length 1, the edges between meshes can be quite long (nearly half the side of the layout). By pulling in meshes closer towards the top level, we can reduce the length of the longest edge considerably. This technique was introduced in [3] to produce minimax edge length layouts for trees, and generalized to graphs with known separators. In the following theorem we lay out the truncated tree of meshes with shorter edges, using a simplified version of the argument introduced in [3]. This layout will later be used to find layouts with short edges for graphs embedded within the truncated tree of meshes.

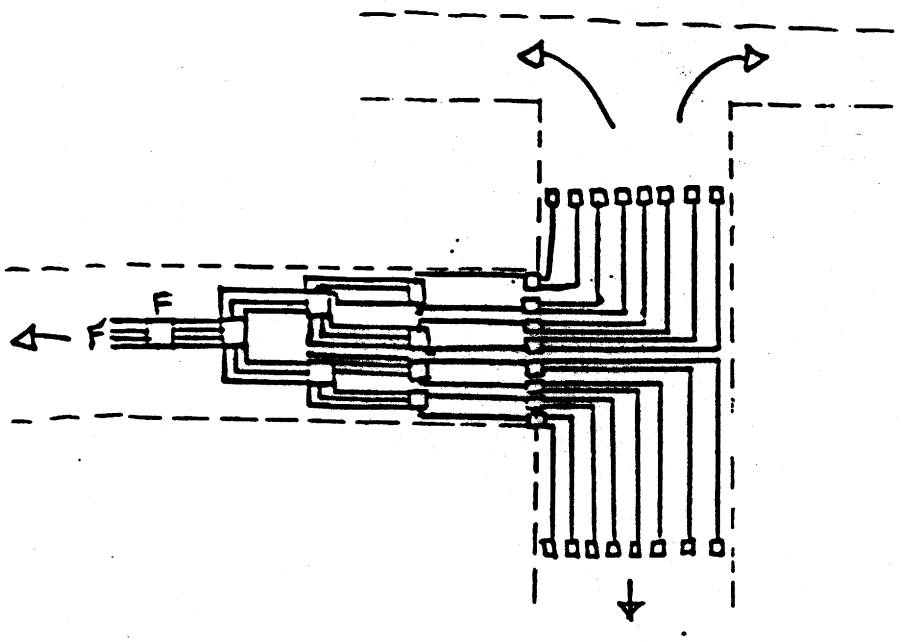

Figure 8. *An improved layout for the tree of meshes.*

Theorem 10. *The truncated tree of meshes $T_{F,2\log\frac{N}{F}}$ can be laid out in area $O(F^2\log^2\frac{N}{F})$ so that mesh edges have length 1 and edges between meshes have length at most $O(F\log\frac{N}{F}/\log\log\frac{N}{F})$.*

Proof. Consider the H-tree layout of a complete binary tree of height $2\log\log\log\frac{N}{F}$, and having $(\log\log\frac{N}{F})^2$ leaves. Expand each linear dimension by a factor $\beta = \Theta(F\log\frac{N}{F}/\log\log\frac{N}{F})$, so that each edge of the H-tree layout becomes a channel of width β and each node becomes a $\beta\times\beta$ square. The resulting area is $(\beta\log\log\frac{N}{F})^2 = \Theta(F^2\log^2\frac{N}{F})$.

Since the channels are much wider than the side of any mesh, we can stack many meshes within one channel. In particular, as seen in Figure 8, we embed the top level mesh at the center of the layout with the second-level meshes on either side. In the first stage of the layout, the meshes in the top levels are placed together in a breadth-first manner. Meshes at successive levels are equally spaced at distance $\Theta(F\log\frac{N}{F}/\log\log\frac{N}{F})$ apart.

We need to ensure that every channel is wide enough to accomodate the meshes stacked within it. To this end, let us suppose that all meshes embedded in the first stage are stacked together in the same channel. Of course, this is a gross overestimate, but suffices for our argument. Since the path from the root to a leaf in the original $(\log\log\frac{N}{F})^2$-leaf H-layout has length $\Theta(\log\log\frac{N}{F})$, a total of $c\log\log\frac{N}{F}$ levels of $T_{F,2\log\frac{N}{F}}$ are embedded in the first stage. The value of the constant c depends on the values of the other constants in the Θ-terms and can be made as small as necessary.

The total number of meshes embedded in the first stage is no more than $2^{1+c\log\log\frac{N}{F}}$. Each mesh has side length no greater than F, so to stack all these meshes within one channel of side β, it suffices to have:

$$F2^{1+c\log\log\frac{N}{F}} \leq O\left(\frac{F\log\frac{N}{F}}{\log\log\frac{N}{F}}\right),$$

which is easily satisfied when $c \leq 1/2$. Hence every channel has sufficient width to stack all the ith level meshes across the channel for any $i \leq c\log\log\frac{N}{F}$.

In the second stage, we embed the remaining meshes in the $\beta \times \beta$ squares. A total of $(\log\frac{N}{F})^c/(\log\log\frac{N}{F})^2$ copies of an $O(\log\frac{N}{F})$ level $\frac{F}{(\log\frac{N}{F})^{c/2}} \times \frac{F}{(\log\frac{N}{F})^{c/2}}$ truncated tree of meshes must be embedded in each of the $(\log\log\frac{N}{F})^2$ $\beta \times \beta$ regions to accomplish this. Using the layout described in Theorem 9 for each copy, the total area required in each region is

$$\Theta\left(\frac{(\log\frac{N}{F})^c}{(\log\log\frac{N}{F})^2}\frac{F^2}{(\log\frac{N}{F})^c}\log^2\left(\frac{N}{F}\right)\right) = \Theta\left(\frac{F^2\log^2\frac{N}{F}}{(\log\log\frac{N}{F})^2}\right).$$

This is precisely the amount of area available in each $\beta \times \beta$ region. Hence the embedding is possible.

It remains to verify that the edges between meshes have length $O(F\log\frac{N}{F}/\log\log\frac{N}{F})$. This is easily done since meshes in adjacent levels were spaced distance $\Theta(F\log\frac{N}{F}/\log\log\frac{N}{F})$ apart in the first stage, and since meshes in adjacent levels were located in the same $\beta \times \beta$ region in the second stage. ∎

6. Solutions to the Eight Problems

Using the framework described in the previous section, we are now ready to present general solutions to the eight problems posed in Section 3. Not surprisingly, the methods of the previous section apply almost directly to these diverse problems. This supports the belief that the divide-and-conquer strategy based on bifurcators is an efficient paradigm for VLSI graph layout, and that the tree of meshes is a versatile network for solving layout problems. The solutions presented in this section are evaluated by comparing them with lower bounds. Some of the lower bounds are new; to maintain continuity, their proofs are deferred to Section 8.

The first two problems, concerning area-efficient layouts and minimax edge length layouts, were already addressed directly in the previous section.

Problem 1. *Given a graph G, produce an area-efficient layout for G.*

By Theorem 7 in Section 5.3, every N-node graph with an $(F, \sqrt{2})$-bifurcator can be embedded in the truncated tree of meshes $T_{O(F),2\log\frac{N}{F}}$. Next, by Theorem 9 in Section 5.4, the truncated tree of meshes can be laid out in $O(F^2\log^2\frac{N}{F})$ area. Therefore, every N-node graph with an $(F, \sqrt{2})$-bifurcator can be laid out in $O(F^2\log^2\frac{N}{F})$ area.

As a simple consequence of Lemma 4, every N-node graph whose smallest $\sqrt{2}$-bifurcator is F, *must* occupy at least F^2 area. For otherwise the graph would have a $\sqrt{2}$-bifurcator strictly smaller than F. Therefore, for *every* graph the upper bound is at most a factor of $O(\log^2\frac{N}{F})$ worse than optimal. As we shall see in Section 8, the upper bound is also *existentially* optimal in that there are N-node graphs with $(F, \sqrt{2})$-bifurcators for all N and F with minimum area $\Omega(F^2\log^2\frac{N}{F})$.

Special Cases. Graphs with $(F, \sqrt{2})$-bifurcators with either of the special forms described in Section 5.1.1 have $O(F^2)$-area layouts.

Problem 2. *Given a graph G, produce an area-efficient layout for G with minimax edge length.*

From Theorem 8 we know that every N-node graph with an $(F, \sqrt{2})$-bifurcator can be embedded in the truncated tree of meshes $T_{O(F),2\log\frac{N}{F}}$ so that no edge passes through more than a constant number of intermediate meshes. Furthermore, the layout for the truncated tree of meshes given in Theorem 10 guarantees that every edge between meshes has length bounded by $O(F\log\frac{N}{F}/\log\log\frac{N}{F})$, and that every edge within a mesh has length one. Combining these two theorems, we see that every N-node graph with an $(F, \sqrt{2})$-bifurcator has an $O(F^2\log^2\frac{N}{F})$-area layout with maximum edge length bounded by $O(F\log\frac{N}{F}/\log\log\frac{N}{F})$.

This bound is also existentially optimal, as will be seen in Section 8. However, the bounds are not guaranteed to be universally close. The only general lower bound on minimax edge length for N-node graphs whose minimum $\sqrt{2}$-bifurcator is F, is $\Omega(F^2/N)$. (This lower bound is also existentially optimal, as will be shown in Section 8.)

The problem of minimizing maximum edge length appears to quite difficult. Although the preceding bounds are disappointingly weak, they are the best known. Bhatt and Cosmadakis [2] show that even determining if a tree can be laid out with minimax edge length one, is NP-complete.

Special Cases. The minimax edge length bounds for graphs with special $(F, \sqrt{2})$-bifurcators are $O(\sqrt{N}/\log N)$ for type A $\sqrt{2}$-bifurcators and $O(F)$ for type B $\sqrt{2}$-bifurcators.

Problem 3. *Given a graph, produce an area-efficient layout in which each wire has bounded delay in the capacitive model.*

First we formalize some details of the model. As usual, a graph describes a connection of processors, with an edge corresponding to a bidirectional link between two processors. Each node is a processing element which contains one driver and one receiver for each incident edge. Every transistor in a processing element has the same size. Thus, in our layouts, a node may be represented by a long and skinny box of constant thickness, with length equal to the area of an internal transistor. Since each node has bounded degree, a box will be just big enough to contain all the transistors in the corresponding processor. Note that different nodes in the layout will have different lengths, but the same thickness. We assume that the grid spacing is adjusted so that nodes and edges have unit thickness and may be laid along grid lines. Although wires are allowed to cross, we will not allow nodes to cross; this corresponds to transistors not overlapping. Similarly, wires and nodes may not cross. The propagation delay over a wire of length l driven by a transistor of area D with capacitive load A is proportional to $(l+A)/D$. The capacitive load presented to a transistor equals the sum of incident wire lengths and areas of adjacent transistors.

Theorem 11. *Every N-node graph G with an $(F, \sqrt{2})$-bifurcator has a bounded-delay layout of area $O(F^2\log^2\frac{N}{F})$.*

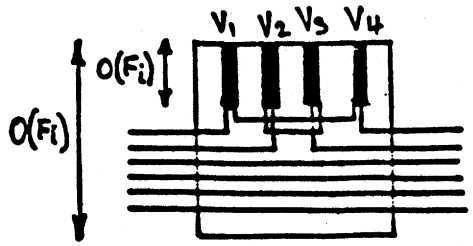

Figure 9. *Laying out expanded nodes in a mesh.*

Proof. As in Theorem 8 of Section 5.4, embed G in a tree of meshes so that adjacent nodes are mapped to meshes no more than a constant number of levels apart. Since the dimensions of meshes at successive levels, as well as the lengths of edges connecting adjacent meshes in the layout of Theorem 9, decrease at the same geometric rate, we know that the length of an edge of G is proportional to the side lengths of the meshes that contain the corresponding nodes. Assign to each node an area that is proportional to the side lengths of the mesh in which it is embedded. Thus, the capacitive load on any node, which equals the sum of the areas of all the incident edges and adjacent nodes, is proportional to the area of the node. In other words, every wire in the layout has bounded delay.

We need to ensure that each enlarged node can be accomodated in its assigned mesh without blowing up the area of the layout by more than a constant factor. This can be done by increasing the dimensions of each mesh by a constant factor, and laying out the nodes and incident edges as shown in Figure 9. Notice that the nodes do not overlap other nodes or wires. The area of each node remains proportional to the side lengths of the mesh containing it, and thus the delay across every wire is bounded. ∎

Special Cases. Similarly, graphs with special $(F, \sqrt{2})$-bifurcators have $O(F^2)$-area bounded-delay layouts.

Theorem 11 means that the area bounds for bounded-delay layouts are no worse than the best known *general* area bounds described for Problem 1. However, it is not known whether or not there exists a graph for which any bounded-delay layout requires asymptotically greater area than the minimum area layout. In the following corollary, we show that the required increase in area is not very large.

Corollary 12. *Any layout of area A for an N-node graph can be transformed into a bounded-delay layout of area $O(A \log^2 \frac{\sqrt{A}}{N})$.*

Proof. By Lemma 4 of Section 5.1, every graph with a layout of area A has a $(\sqrt{A}, \sqrt{2})$-bifurcator which can be quickly found. Then by Theorem 11, we can construct a bounded-delay layout with area $O(A \log^2 \frac{\sqrt{A}}{N})$. ∎

Remark. Unlike the previous area bounds which can be obtained only when the bifurcator for a graph is already known, the preceding corollary for transforming a layout into a bounded-delay layout can be efficiently implemented.

Problem 4. *Given a graph G, produce a layout for G with few wire crossings.*

The layouts for the truncated tree of meshes in Theorems 9 and 10 do not have any edge crossings. Since every N-node graph G with an $(F, \sqrt{2})$-bifurcator can be embedded within the truncated tree of meshes $T_{O(F),2\log\frac{N}{F}}$, this means that the number of crossings in the layout for G cannot exceed the number of nodes in $T_{O(F),2\log\frac{N}{F}}$. In other words, the number of crossings in the layout for G is bounded by $O(F^2 \log \frac{N}{F})$.

In Section 8 we will see that this bound too is existentially optimal. We will also show that for every N-node graph with a minimum $\sqrt{2}$-bifurcator of size F, the number of crossings plus the number of nodes is at least $\Omega(F^2)$. Thus, if F is asymptotically greater than $\sqrt{N}$, the number of crossings in the layout for G is no worse than a factor $O(\log \frac{N}{F})$ times optimal.

Special Cases. Graphs with special $(F, \sqrt{2})$-bifurcators can be laid out with $O(F^2)$ crossings.

Problem 5. *Given a graph, produce an area-efficient regular layout for the graph.*

In Theorem 7, we showed how to embed any N-node graph G with an $(F, \sqrt{2})$-bifurcator in $T_{cF,2\log\frac{N}{F}}$ for some constant c. Moreover, the nodes of G were divided evenly among the N^2/F^2 bottom-level meshes of $T_{cF,2\log\frac{N}{F}}$ and in each bottom-level mesh, the nodes of G were embedded in a regular fashion. Thus to produce an $O(F^2 \log^2 \frac{N}{F})$-area layout for G that is regular, we need only produce a layout for $T_{cf,2\log\frac{N}{F}}$ for which the nodes at the $(2\log\frac{N}{F})$th level are located in a regular fashion. In fact, we can do much better, as we show in the following theorem.

Theorem 13. *The truncated tree of meshes $T_{O(F),2\log\frac{N}{F}}$ can be laid out in $O(F^2 \log^2 \frac{N}{F})$ area so that, for every level i, all nodes within ith level meshes are placed in a regular fashion.*

Proof. The first step is to construct a $\Theta(\log \frac{N}{F})$-layer three-dimensional layout [23] of the truncated tree of meshes. Fold the connections between the root of the tree of meshes and each of its two sons so that the sons fit naturally on a second layer over the root mesh. Fold the connections to each of the meshes at the next lower level so they fit, on the third layer, directly over the meshes on the second layer, and so forth. This generates a $\log \frac{N}{F}$-layer three-dimensional layout, with each layer occupying linear area. By projecting the three-dimensional layout onto the plane in the manner of Thompson [42, pp. 36-38], the result follows. (The same layout can be constructed by interleaving the meshes at each level.) ∎

Special Cases. The $O(F^2)$-area layouts for graphs with special $\sqrt{2}$-bifurcators are also regular.

Problem 6. *Design area-efficient chips that can be configured to realize a large number of graphs.*

In section 5.3 we showed that every N-node graph with an $(F, \sqrt{2})$-bifurcator can be embedded in a truncated tree of meshes such that the nodes of the graph are embedded in a regular fashion in the bottom-level meshes of $T_{cF,2\log\frac{N}{F}}$. In fact, the nodes can be mapped to fixed positions within the meshes. Therefore, if we lay out the truncated tree of meshes on a chip with processors at these fixed positions, we have a configurable chip for all graphs with the corresponding bifurcator. This yields the following result. Observe that the area bounds for configurable layouts are the same as for unrestricted layouts.

Theorem 14. *Every N-node graph with an $(F, \sqrt{2})$-bifurcator has a configurable layout of area $O(F^2 \log^2 \frac{N}{F})$.*

Proof. Simply make the connections in the meshes after the rest of the chip has been fabricated. Recall that we used the meshes as crossbar switches in Theorem 7. ∎

Special Cases. Similarly, graphs with special bifurcators have $O(F^2)$-area configurable layouts.

Problem 7. *On a wafer which has arbitrarily distributed defective cells, realize a given graph on the good cells.*

In Section 5.3 (Theorem 7) we showed how to embed any N-node graph G with an $(F, \sqrt{2})$-bifurcator in the truncated tree of meshes $T_{O(F),2\log\frac{N}{F}}$. The embedding had the property that nodes of the graph could be mapped to fixed positions within the meshes at the bottom level. Accordingly, we fixed processors at each of these positions.

Faulty processors on a wafer therefore correspond to faulty processors in the truncated tree of meshes, the correspondence being induced via the layout for the tree of meshes. It is clearly no longer possible to realize G in the faulty tree of meshes. However, it is possible to realize a smaller graph with a similar structure using only the functioning processors.

More formally, consider a class of graphs for which any N-node graph in the class has a $\sqrt{2}$-bifurcator of size $O(f(N))$ where the function f is such that $f(x)/\sqrt{x}$ is nondecreasing for increasing x. For example, $f(x) = \sqrt{x}$ for the class of square meshes (as well as for the class of trees or the class of planar graphs). In what follows, we will show how to embed any M-node graph from the class in any $T_{cf(N),2\log\frac{N}{f(N)}}$ that has M functioning processors where $N \geq M$ and c is a sufficiently large constant. In particular, we will show how to embed $T_{f(M),2\log\frac{M}{f(M)}}$ in the structure. By the results in Section 5.3 of the paper, this will be sufficient to prove the claim. Thus the layout strategy developed in Section 5 is impervious to the existence of faulty processors. This result substantially generalizes and simplifies a similar result proved by Leighton and Leiserson for embedding meshes around faults in [22].

Theorem 15. *Given the preceding constraints on N, M, c and f, a completely functioning truncated tree of meshes $T_{f(M),2\log\frac{M}{f(M)}}$ with M processors can be embedded in any partially functioning truncated tree of meshes $T_{cf(N),2\log\frac{N}{f(N)}}$ with N processors (M of which are functioning) so that the processors of the former are mapped onto the functioning processors of the latter.*

Proof. Label the functioning processors in each tree of meshes from 1 to M by counting from left to right across the bottom level of each graph. (Recall that the processors are evenly distributed on the bottom level.) Map the kth processor of $T_{f(M),2\log\frac{M}{f(M)}}$ onto the kth functioning processor of $T_{cf(N),2\log\frac{N}{f(N)}}$. Route the edges of the former graph through the meshes of the latter in the usual way, at the same time embedding meshes of the former in blocks within the meshes of the latter.

It remains to show that the capacity of each mesh in $T_{cf(N),2\log\frac{N}{f(N)}}$ is sufficient for the embedding. Consider a mesh X on the ith level of $T_{cf(N),2\log\frac{N}{f(N)}}$. This mesh has side lengths $cf(N)/2^{i/2}$ and at most $N/2^i$ functioning processors below it in the bottom level of the graph. The only meshes and edges of $T_{f(M),2\log\frac{M}{f(M)}}$ that are embedded in X are those that correspond to roots of the forest of complete binary trees formed by removing the corresponding interval of (at most $N/2^i$) processors in $T_{f(M),2\log\frac{M}{f(M)}}$. These roots are identified by splitting $T_{f(M),2\log\frac{M}{f(M)}}$ (as in Lemma 3) at the two endpoints of the interval. There are at most two roots at each level in the resulting forest and the sum of their side lengths (a geometrically decreasing sum) is proportional to $f(M)/2^{j/2}$ where j is such that $M/2^j \leq N/2^i$. (Remember that there are at most $N/2^i$ processors in the leaves of the forest so that the height of the largest complete binary tree in the forest is j where $M/2^j \leq N/2^i$.) Thus the sum of the side lengths of the meshes embedded in X is $O\left(\frac{f(M)}{2^{i/2}}\sqrt{\frac{N}{M}}\right)$ which, for sufficiently large c, is less than $cf(N)/2^{i/2}$ (this is the side length of X), since $N \geq M$ and $f(x)/\sqrt{x}$ is a nondecreasing function. Hence X is large enough and the embedding is possible. ∎

Special Cases. A similar argument works for graphs with special bifurcators.

Problem 8. *Given a graph G, assemble G using the minimum number of copies of a single chip having few external pin connections.*

Suppose that we wish to assemble N-node graphs with $(F,\sqrt{2})$-bifurcators but that each chip contains only m nodes, where $m < N$. Consider a chip consisting of a truncated tree of meshes $T_{O(\frac{\sqrt{m}F}{\sqrt{N}}),O(\log\frac{\sqrt{mN}}{F})}$, with the m processors divided equally among the bottom-level meshes, and external pin connections to the top of the top level mesh. Two copies of this chip may be wired together to form a truncated tree of meshes with $2m$ processors. Thus, graphs with twice as many processors can be assembled with two chips than can be assembled on a single chip. More generally, we have the following result.

Theorem 16. *There is a universal restructurable chip with m processors and $O(\frac{\sqrt{m}F}{\sqrt{N}})$ external pins, occupying area $O(\frac{F^2m}{N}\log^2\frac{\sqrt{mN}}{F})$, such that every N-node graph with an $(F,\sqrt{2})$-bifurcator can be assembled using multiple copies of the universal chip. Furthermore, the number of chips used in the assembly is as small as possible.*

Proof. Consider the top $\log N - \log m$ levels of a fully balanced decomposition tree of G. Each of the subgraphs at level $\log N - \log m$ has $N/2^{\log N-\log m} = m$ nodes, and has a $\sqrt{2}$-

bifurcator of size $O(\frac{\sqrt{m}F}{\sqrt{N}})$. By Theorem 7, each of these subgraphs can be realized with a single universal chip consisting of a truncated tree of meshes $T_{O(\frac{\sqrt{m}F}{\sqrt{N}}),O(\log \frac{\sqrt{mN}}{F})}$ whose area is bounded by $O(\frac{F^2m}{N}\log^2 \frac{\sqrt{mN}}{F})$, and which has $O(\frac{\sqrt{m}F}{\sqrt{N}})$ external pin connections. To complete the assembly, the chips are wired up by making connections between pins on different chips as given by the decomposition tree. ∎

A noteworthy consequence of this result is that when $F = O(\sqrt{N})$, the restructurable chip has $O(\sqrt{m})$ pins, which is independent of the size of the network to be assembled. This is the best possible. To realize networks with larger bifurcators, the parameters of the restructurable chip depend on the size of the network assembled.

Special Cases. For graphs with special bifurcators, the same is true except that only $O(F^2)$ area is used on each chip. For type A $\sqrt{2}$-bifurcators, the number of pins needed is much lower. For example, N-node trees require only $O(\log m)$ pins per chip [4]. (As is the case for all planar graphs, the number of pins does not depend on the number of nodes. This is because N-node planar graphs have $\sqrt{2}$-bifurcators of size $O(\sqrt{N})$.) Recently, we improved this result to 6 pins for trees by using slightly different techniques (but by giving up the use of a small portion of the processors on some chips). Hence, pin count constraints place no limit at all on the size of trees that can be fabricated with a single configurable chip, no matter how many processors are placed on each chip.

7. Layout Algorithms Based on Graph Bisection Heuristics

In the previous section we saw how a variety of layout problems could be efficiently solved once the decomposition tree of a graph was known. All the results were of the flavor: "If G has an $(F, \sqrt{2})$-bifurcator, then ...." But, given a graph, how do we find a small $\sqrt{2}$-bifurcator or a suitable decomposition tree for the graph?

Some graphs are easy to decompose, so that a small bifurcator can be found relatively easily. Such graphs include trees, cube-connected cycles, and, more generally, graphs that are constructed recursively. It is also easy to find a small bifurcator if a small-area layout is known. (From Lemma 4, recall that graphs with layout area A have a $(\sqrt{A}, \sqrt{2})$-bifurcator.)

In general however, it is extremely difficult to find small bifurcators for graphs. The reason is that the process of graph decomposition involves the problem of graph partitioning, or graph bisection. The graph bisection problem, also known as the "min-cut" problem, requires a graph to be partitioned into two components of equal size, removing the minimum possible number of edges. This problem is known to be NP-complete [13].

There are, however, a large number of heuristics for bisecting graphs which appear to perform well in practice [6, 7, 10, 18, 37, 40]. Many automated layout systems use these and other partitioning heuristics. Is there any theoretical justification for this? In what follows, we answer affirmatively by showing that a provably good algorithm for graph bisection can be tailored into a provably good layout algorithm.

The key idea is to convert a bisection width heuristic into a heuristic for drawing graphs

with few crossings. (Determining the crossing number is also NP-Complete [14].) Like small-area layouts, such drawings can be used to find small $\sqrt{2}$-bifurcators. The following theorem shows that with a provably good bisection heuristic, the number of crossings is provably small (i.e., within guaranteed bounds from optimal).

Theorem 17. *Suppose there is an algorithm which, for every N-node graph with bisection width B, finds a bisection of size at most $\gamma(N)B$ in polynomial time. ($\gamma(N)$ is some nondecreasing functional measure of error.) Then there is a polynomial time algorithm which, for every N-node graph with crossing number C, produces a drawing with at most $O((C+N)\gamma^2(N)\log^2 N)$ crossings.*

Proof. Use the bisection width algorithm to produce a decomposition tree for G by recursively bisecting each subgraph in the tree. As in Figure 4, define G_{w0} and G_{w1} to be the left and right sons of G_w in the decomposition tree. Further define B_w to be the bisection width of G_w, C_w to be the crossing number of G_w and N_w to be the number of nodes in G_w. Clearly, $N_w = N/2^{|w|}$. A simple application of the planar separator theorem shows that $C + N \geq \Omega(B^2)$ for any graph and thus $C_w + N_w \geq \Omega(B_w^2)$ for every w [19, 20]. Since G_w contains $G_{ww'}$ for every w', we also know that $C_w \geq C_{ww'}$ and thus that $C_w + N_w \geq \Omega(B_{ww'}^2)$ for every w'.

The algorithm for drawing G is recursive. At each step, we will use drawings of G_{w0} and G_{w1} to construct a drawing of G_w. In addition, we will store a path from each node to the exterior face of the drawing which has a small number of crossings. These paths are used when inserting edges at each recursive step, but are otherwise only remembered and updated (i.e., they do not count in the crossing totals). Let C'_w be the number of crossings in the constructed drawing of G_w and let P_w be the maximum number of edges that would have to be crossed to draw an edge from any node in the constructed drawing of G_w to the exterior of the drawing. Using a straightforward divide-and-conquer analysis similar to that used to prove Theorem 7-8 of [19], we can see that

$$C'_w \leq C'_{w0} + C'_{w1} + \gamma^2(N)B_w^2 + \gamma(N)B_wP_w$$

and

$$P_w \leq \max(P_{w0}, P_{w1}) + \gamma(N)B_w$$

for every w. Solving the latter recurrence, we find that

$$P_w \leq \gamma(N)\max_{w'}(B_{ww'})\log N_w$$

and thus that

$$C'_w \leq C'_{w0} + C'_{w1} + O(\gamma^2(N)(C_w + N_w)\log N_w).$$

It is now a straightforward matter to prove by induction on $|\,w\,|$ (starting with $|\,w\,| = \log N$ and decreasing) that

$$C'_w \leq O(\gamma^2(N)(C_w + N_w)\log^2 N_w),$$

thus proving the theorem. ∎

As a consequence of Theorem 17, we can prove the following result on finding $\sqrt{2}$-bifurcators.

Theorem 18. *If there exists a polynomial time algorithm which finds a $\gamma(N)B$-bisection of any N-node graph with bisection width B, then there is a polynomial time algorithm for finding a $(\rho(N)F, \sqrt{2})$-bifurcator for any graph G where F is the size of the minimum $\sqrt{2}$-bifurcator for G and $\rho(N) = O(\gamma(N)\log^{3/2} N)$.*

Proof. First use Theorem 17 to construct a drawing for G with $C' = O(\gamma^2(N)\log^2 N(C+N))$ crossings where C is the minimal crossing number of G. In what follows, we show how this drawing can be used to construct a $\sqrt{2}$-bifurcator for G of size $O(\gamma(N)\log N\sqrt{C+N})$.

Consider the graph G' formed by replacing the C' edge crossings in the drawing of G with *artificial nodes*. This graph is planar and has $M = N + C'$ nodes. By the Lipton-Tarjan planar separator theorem [29], we can conclude that G' has a $\sqrt{2}$-bifurcator of size $O(\sqrt{M}) = O(\sqrt{N+C'})$. Thus G has a $\sqrt{2}$-bifurcator of size $O(\sqrt{N+C'}) = O(\gamma(N)\log N\sqrt{C+N})$.

By the optimality of C and the solution to Problem 4 in Section 6, we know that $C + N \leq O(F^2 \log \frac{N}{F})$ where F is the size of the minimal $\sqrt{2}$-bifurcator of G. Hence, we have constructed a $\sqrt{2}$-bifurcator for G of size $O\left(\gamma(N)(\log N)\, F\sqrt{\log \frac{N}{F}}\right) = \rho(N)F$ where $\rho(N) = O(\gamma(N)\log^{3/2} N)$. ∎

Although Theorem 18 can be easily applied to the layout area problem, better bisection-width-based bounds can be derived directly from Theorem 17. These bounds are stated in the following theorem.

Theorem 19. *If there exists a polynomial time algorithm that finds a $\gamma(N)B$-bisection for any N-node graph with bisection width B, then there exists a polynomial time algorithm that produces a layout for any N-node graph G with area at most $\psi(N)A$ where A is the minimum layout area of G and $\psi(N) = O(\gamma^2(N)\log^4 N)$.*

Proof. First use the algorithm described in Theorem 17 to find a drawing for G with at most $\phi(N)(C+N)$ crossings where C is the crossing number of G and $\phi(N) = O(\gamma^2(N)\log^2 N)$. Convert the drawing into a planar graph by replacing each crossing with an artificial node as in Theorem 18. Using the algorithm developed by Leiserson [26] and Valiant [45], this graph can be laid out using at most $O(\phi(N)(C+N)\log^2 N)$ area. The construction is completed by replacing the artificial nodes with their original edge crossings. Since $A \geq C + N$, it is clear that the layout has area at most $\psi(N)A$ where $\psi(N) = O(\gamma^2(N)\log^4 N)$. ∎

8. Area, Crossing Number and Edge Length Bounds

In Section 6, we argued that the new framework is *universally* good in the sense that *no* graph with an $(F, \sqrt{2})$-bifurcator has a *much* better layout than that provided by the framework. In this section, we show that the framework is *existentially* optimal inasmuch as there exist graphs with $(F, \sqrt{2})$-bifurcators that are laid out optimally by the framework.

8.1. Universal Bounds

In the following theorem, we characterize the layout area, crossing number and minimax edge length of a graph in terms of its minimal $\sqrt{2}$-bifurcator. Most of the bounds have already been proved but we state them together again for convenience.

Theorem 20. *Let F be the minimum $\sqrt{2}$-bifurcator of an N-node graph G, which has minimum layout area A, minimax edge length L, and crossing number C. The following inequalities hold, and the upper bounds can all be realized simultaneously.*

$$F^2 \leq A \leq O\left(F^2 \log^2 \frac{N}{F}\right),$$

$$\Omega(F^2) \leq C + N \leq O\left(F^2 \log \frac{N}{F}\right)$$

and

$$\Omega(F^2/N) \leq L \leq O\left(F \log \frac{N}{F} \Big/ \log\log \frac{N}{F}\right).$$

Proof. The upper bounds were proved in the solutions to Problems 1, 2 and 4 in Section 6. Note that the bounds are all realized for the same layout.

The area lower bound is from Lemma 4. The crossing number lower bound follows from the analysis in Theorem 18. In particular, any N-node graph with crossing number C has a $\sqrt{2}$-bifurcator of size $O(\sqrt{N+C})$. The edge length lower bound follows from the crossing number lower bound. Since $C + N \geq \Omega(F^2)$, the wire area of the layout is at least that large and thus at least one of the $\Theta(N)$ wires in the network must have length $\Omega(F^2/N)$. (In fact, the average edge length is $\Omega(F^2/N)$.) ∎

As we have noted throughout the paper, it is possible to improve the upper bounds in Theorem 20 for special classes of graphs. As we show in the next section however, such improvements are not always possible.

8.2. Existential Bounds

We next show that the universal upper and lower bounds given in Theorem 20 are everywhere existentially tight. We first define the expander-connected mesh and show that it achieves (simultaneously) the universal lower bounds on area, crossing number and edge length for any N and F. Then we define the expander-connected mesh of trees and show that it attains the corresponding universal upper bounds.

An *expander-connected mesh* $P_{m,n}$ with $N = mn^2$ nodes is formed by superimposing n^2 copies of an m-node expander graph on m copies of an $n \times n$ mesh. More precisely, define $P_{m,n}$ to be the graph consisting of m disjoint n-by-n meshes which are interlinked with additional

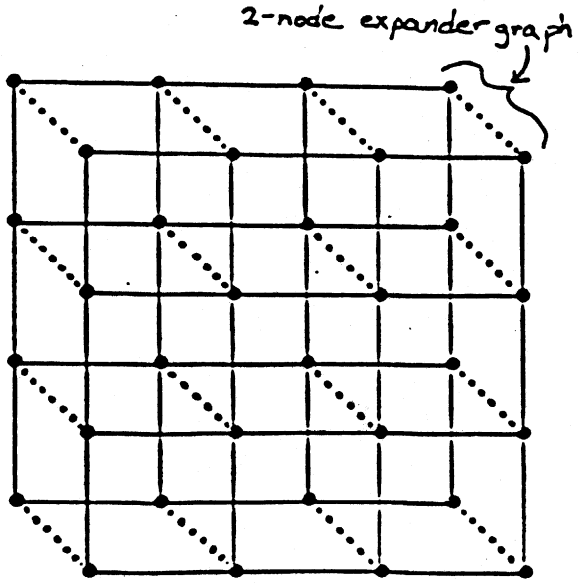

Figure 10. *The expander-connected mesh $P_{2,4}$.*

edges so that for each i and j ($1 \leq i, j \leq n$), the subgraph induced on the m nodes which are in the (i, j) position of some mesh is an expander graph. For example, $P_{2,4}$ is shown in Figure 10. The dotted lines represent edges in the expander graphs while the solid lines represent edges in the meshes.

Remark. Strictly speaking, the expander-connected mesh has node degree 7 and does not fit into our layout model. This problem can be dealt with in a variety of ways but the simplest is to replace each degree 7 node with a 7-leaf binary tree. The area, crossing number and minimax edge length bounds for the resulting degree 3 graph differ by at most a constant factor from those derived below for the unaltered graph. A similar fact is also true for the expander-connected mesh of trees.

In the following we show that the size of the smallest $\sqrt{2}$-bifurcator of $P_{m,n}$ is at least $\Omega(mn)$. This is accomplished using the lower bound techniques developed in [19, 20] to prove that the *bisection width* of $P_{m,n}$ is at least $\Omega(mn)$. This means that the smallest $\sqrt{2}$-bifurcator for $P_{m,n}$ has size $\Omega(mn)$.

Lemma 21. *The bisection width of $P_{m,n}$ is at least $\Omega(mn)$.*

Proof. Let (i, j, k) denote the (i, j) node of the kth mesh of $P_{m,n}$. In addition, let $P'_{m,n}$ denote the graph formed by extending each expander graph of $P_{m,n}$ to a complete graph (i.e., to the graph formed by inserting edges between nodes (i, j, k) and (i, j, k') for every $1 \leq i, j \leq n$ and $1 \leq k, k' \leq m$). In what follows, we will use the methods of [19, 20] to find a lower bound on the bisection width of $P'_{m,n}$. This, in turn, will be used to find a lower bound on the bisection width of $P_{m,n}$.

Consider the embedding of the complete graph on mn^2 nodes (K_{mn^2}) in $P'_{m,n}$ which links node (i,j,k) to node (i',j',k') via the path

$$(i,j,k) \rightarrow (i \pm 1, j, k) \rightarrow (i \pm 2, j, k) \rightarrow \cdots \rightarrow (i', j, k)$$
$$\rightarrow (i', j \pm 1, k) \rightarrow (i', j \pm 2, k) \rightarrow \cdots \rightarrow (i', j', k)$$
$$\rightarrow (i', j', k').$$

(Note that the notion of an embedding used here is different than that defined in Section 2, where edges were mapped to edge-disjoint paths in the grid.)

A simple counting argument reveals that each mesh edge of $P'_{m,n}$ is utilized at most $O(mn^3)$ times by the embedding of K_{mn^2} while each complete graph edge is used at most $O(n^2)$ times. Since at least $m^2n^4/4$ edges of K_{mn^2}, must cross any bisection of K_{m,n^2}, we can thus conclude that any bisection of $P'_{m,n}$ must cut at least $\Omega(mn)$ mesh edges or at least $\Omega(m^2n^2)$ complete graph edges. Clearly, any bisection of $P'_{m,n}$ which cuts $\Omega(mn)$ mesh edges must also cut $\Omega(mn)$ mesh edges of $P_{m,n}$. In what follows, we will show that any bisection of $P'_{m,n}$ which cuts s complete graph edges must cut at least $\Omega(s/m)$ expander edges of $P_{m,n}$. This will imply that any bisection of $P'_{m,n}$ which cuts $\Omega(m^2n^2)$ complete graph edges must cut $\Omega(mn^2)$ expander graph edges of $P_{m,n}$, thus completing the proof.

Consider a bisection of $P'_{m,n}$ which cuts s complete graph edges. Let $s_{i,j}$ denote the number of edges cut in the (i,j) complete graph of $P'_{m,n}$ for $1 \leq i,j \leq n$. Clearly, $s = \sum_{i,j=1}^{n} s_{i,j}$. As each node in an m-node complete graph is incident to at most $m-1$ edges, we know that the bisection of $P'_{m,n}$ divides the (i,j) complete graph into two subgraphs which contain at least $s_{i,j}/m$ nodes each. Thus at least $\Omega(s_{i,j}/m)$ edges of the (i,j) expander graph of $P_{m,n}$ are cut by the bisection. Summing, we find that the bisection cuts at least $\Omega(s/m)$ expander edges of $P_{m,n}$ in total. ∎

We can construct an expander-connected mesh with N nodes and minimum $\sqrt{2}$-bifurcator F for any N and F such that $\Omega(\sqrt{N}) \leq F \leq O(N)$, by setting $n = \Theta(N/F)$ and $m = \Theta(F^2/N)$. We now show how to construct a layout for $P_{m,n}$ which achieves (up to a constant) the universal lower bounds for area, crossing number and minimax edge length of Theorem 20.

Theorem 22. *There is a layout for $P_{m,n}$ which has area and crossing number at most $O(m^2n^2) = O(F^2)$ and maximum edge length at most $O(m) = O(F^2/N)$.*

Proof. Lay out each expander graph in an $O(m)$-by-$O(m)$ grid so that the node in the kth mesh is in the (k,k) position of the grid. Arrange these sublayouts in a mesh-like pattern so as to be consistent with the mesh structure of $P_{m,n}$. Next insert the mesh edges in the natural way. The resulting layout should look like Figure 10. It is easily verified that the area of this layout (and hence its crossing number) is at most $O(n^2) \times O(m^2) = O(m^2n^2)$, and that every edge has length at most $O(m)$. ∎

Before defining the expander-connected mesh of trees, it is useful to review the definition of a mesh of trees as proposed by Leighton in [19, 20]. (An equivalent structure, the *orthogonal trees network*, has been studied by Nath, Maheshwari and Bhatt in [33]. Cappello and Stieglitz have

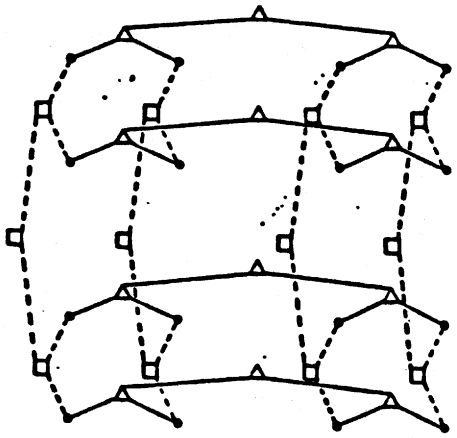

Figure 11. *The* 4×4 *mesh of trees* $M_{2,4}$.

also studied this graph, which they call the *orthogonal forests*, in [8].) The *2-dimensional mesh of trees* $M_{2,n}$ (where n is assumed to be a power of 2) is defined as follows. Starting with an $n \times n$ matrix of nodes and adding nodes wherever necessary, construct a complete binary tree in every row and column of the matrix. The trees should be constructed so that

- the leaves in each tree are precisely the nodes in the corresponding row or column of the original matrix, and
- the subgraph induced on the nodes in each quadrant is $M_{2,n/2}$.

For example, we have drawn $M_{2,4}$ in Figure 11. The nodes in the original 4×4 matrix are represented by dots. The nodes which were added in order to form row trees are drawn as small triangles while those added to form column trees are shown as small squares. Solid lines indicate row tree edges while dashed lines indicate column tree edges.

The expander-connected mesh of trees is similar to the expander-connected mesh $P_{m,n}$ except that the meshes are replaced by meshes of trees. More precisely, the *expander-connected mesh of trees* (denoted by $Q_{m,n}$) is defined to be the graph consisting of m disjoint $n \times n$ meshes of trees which are interlinked with additional edges so that for each i and j $(1 \leq i, j \leq n)$, the subgraph induced on those *leaves* in the (i, j) position of some mesh of trees is an expander graph. For example, we have drawn $Q_{2,2}$ in Figure 12. The dotted lines represent edges in the expander graphs while the dashed and solid lines represent edges in the meshes of trees.

It is not difficult to check that $Q_{m,n}$ has $N = \Theta(mn^2)$ nodes and a $\sqrt{2}$-bifurcator of size $F = mn$. In the following theorem, we will show that $Q_{m,n}$ has layout area at least $\Omega(m^2n^2 \log^2 n) = \Omega(F^2 \log^2 \frac{N}{F})$, crossing number at least $\Omega(m^2n^2 \log n) = \Omega(F^2 \log \frac{N}{F})$ and minimax edge length at least $\Omega(mn \log n / \log\log n) = \Omega(F \log \frac{N}{F} / \log\log \frac{N}{F})$. Thus the universal upper bounds proved in Theorem 20 are existentially tight for every N and F.

Theorem 23. *The expander-connected mesh of trees has layout area* $\Theta(m^2n^2 \log^2 n)$, *crossing number* $\Theta(m^2n^2 \log n)$ *and minimax edge length* $\Theta(mn \log n / \log\log n)$.

Proof. The upper bounds follow trivially from Theorem 20 and the fact that $Q_{m,n}$ has a $\sqrt{2}$-bifurcator of size $O(mn)$. The lower bounds are substantially more difficult. In fact, we suggest that the reader be familiar with the lower bound techniques described in [19, 20] for the case when $m = 1$ before wading through the following proof for general m. We commence with the area lower bound.

8.2.1. Area Bound

Let $W_m(n)$ denote the minimum *wire area* of $Q_{m,n}$. We will show that for a sufficiently small (but positive) constant α,

$$W_m(n) \geq \alpha m^2 n^2 \log^2 n$$

for all m and n. This will, of course, imply the desired lower bound for layout area.

The proof is by induction of n. Since $Q_{m,n}$ contains n^2 disjoint m-node expander graphs, the hypothesis is clearly true for $n \leq 16$ provided that α is a sufficiently small constant. In what follows, we will assume that the hypothesis is true for all values less than n in order to prove it for n.

Consider any layout for $Q_{m,n}$ which uses $W_m(n)$ wire. Partition the layout into three vertical strips V_0, V_1 and V_2 so that the center strip contains $7mn^2/8$ leaves and each outer strip contains $mn^2/16$ leaves. Similarly partition the layout into three horizontal strips H_0, H_1 and H_2 so that the middle strip contains $7mn^2/8$ leaves and each outer strip contains $mn^2/16$ leaves. For example, see Figure 13.

Let d denote the length of the longest side of the center block formed by the intersection of V_1 and H_1. Without loss of generality, we assume that the longest side is horizontal. In what follows, we will show that $d \geq \frac{1}{16}\sqrt{\alpha}mn \log n$.

Since each of the regions $V_0 \bigcap H_1$ and $V_2 \bigcap H_1$ can contain at most $mn^2/16$ leaves, it is clear that $V_1 \bigcap H_1$ contains at least $3mn^2/4$ leaves. Consider the $n^{3/2}$ subgraphs of $Q_{m,n}$ produced by eliminating the top $\frac{3}{4}\log n$ levels of the row and column trees of $Q_{m,n}$. Each of these subgraphs

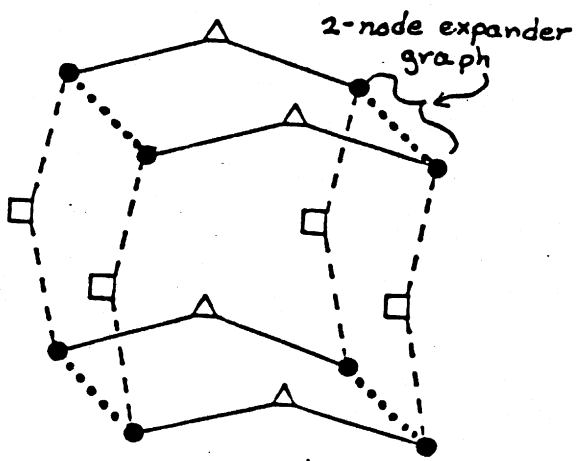

Figure 12. *The expander-connected mesh of trees* $Q_{2,2}$.

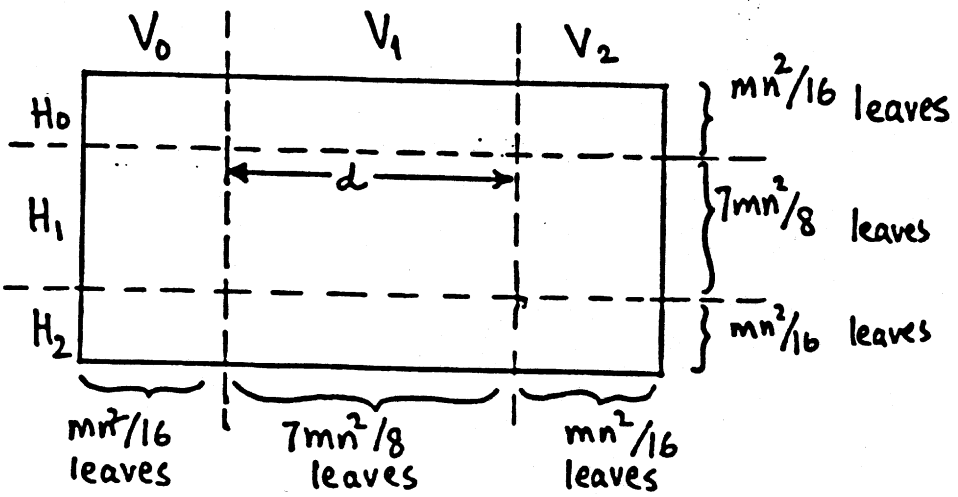

Figure 13. *Partitioning a layout.*

is isomorphic to $Q_{m,n^{1/4}}$. By the pigeonhole principle, at least 1/4 of these subgraphs have at least 1/2 of their leaves inside $V_1 \bigcap H_1$. If $d < \frac{1}{16}\sqrt{\alpha}mn\log n$ (otherwise, we are done), then at most $4d < \frac{1}{4}\sqrt{\alpha}mn\log n$ edges can cross the boundary of $V_1 \bigcap H_1$. Thus, at most $\frac{1}{4}c_0\sqrt{\alpha}n\log n$ of the subgraphs which have most of their leaves in $V_1 \bigcap H_1$ can have m or more nodes or parts of edges outside of $V_1 \bigcap H_1$. (This is because every partition of $Q_{m,n^{1/4}}$ for $n \geq 16$ into two subsets, each of which contains m or more nodes, requires the removal of at least m/c_0 edges where c_0 is a constant.)

This means that $V_1 \bigcap H_1$ contains at least $\frac{1}{4}n^{3/2} - \frac{1}{4}c_0\sqrt{\alpha}n\log n$ nearly complete copies of $Q_{m,n^{1/4}}$. Since (by induction), $W_m(n^{1/4}) \geq \frac{1}{16}\alpha m^2 n^{1/2}\log^2 n$, and since each nearly complete copy of $Q_{m,n^{1/4}}$ is missing at most m nodes and edges, it is not difficult to show that the wire area of each nearly complete copy of $Q_{m,n^{1/4}}$ is at least $\frac{1}{32}\alpha m^2 n^{1/2}\log^2 n$. Thus $V_1 \bigcap H_1$ contains at least

$$(\frac{1}{4}n^{3/2} - \frac{1}{4}c_0\sqrt{\alpha}n\log n) \times \frac{1}{32}\alpha m^2 n^{1/2}\log^2 n$$

wire area. For constant α sufficiently small, this is at least $\frac{1}{256}\alpha m^2 n^2 \log^2 n$. Hence $d \geq \frac{1}{16}\sqrt{\alpha}mn\log n$, as claimed.

We next use the layout for $Q_{m,n}$ to construct a drawing for the complete graph on mn^2 nodes (namely, the mn^2 leaves of $Q_{m,n}$). In particular, the edge from leaf (i,j,k) to leaf (i',j',k') is drawn from (i,j,k) to (i',j',k) along the path from (i,j,k) to (i',j,k) in the jth row tree of the kth tree of meshes and from (i',j,k) to (i',j',k) in the i'th row tree of the kth tree of meshes. The edge to (i',j',k') is completed by drawing a line from (i',j',k) to (i',j',k') directly. (Notice that we have traced over the mesh of trees edges but not the expander edges.) No matter how the edges are drawn in the plane, however, (e.g., they may cross or overlap) it is clear from

Figure 13 that the sum of the lengths of the edges (as measured in Euclidean space) is at least $(mn^2/16)^2 d \geq 2^{-12}\sqrt{\alpha}m^3n^5 \log n$. This is due to the fact that $(mn^2/16)^2$ edges pass from region V_0 to region V_2 and that these regions are separated by a distance d.

Let L_i denote the sum of the lengths of the edges in the ith levels of the binary trees in the layout of $Q_{m,n}$. In addition, let R denote the sum $\sum_{i,j=1}^{n} R_{i,j}$ where $R_{i,j}$ is the sum over $1 \leq k, k' \leq m$ of the distance between (i,j,k) and (i,j,k'). Each level i edge is traced over at most $mn^3 2^{-i}$ times in the drawing of the complete graph. In addition, the straight-line path between (i,j,k) and (i,j,k') is traced over at most n^2 times for any i, j, k and k'. Thus,

$$Rn^2 + \sum_{i=1}^{\log n} L_i n^3 m 2^{-i} \geq 2^{-12}\sqrt{\alpha}m^3n^5 \log n.$$

This means that one of the following inequalities must be true:

$$R \geq 2^{-13}\sqrt{\alpha}m^3n^3 \log n$$

or

$$\sum_{i=1}^{\log n} L_i 2^{-i} \geq 2^{-13}\sqrt{\alpha}m^2n^2 \log n.$$

In the first case, we observe that there is a constant c_1 such that $R' \geq \frac{c_1}{m}R$ where $R' = \sum_{i,j=1}^{m} R'_{i,j}$ and $R'_{i,j}$ is the sum of the lengths of the edges in the (i,j) expander graph of $Q_{m,n}$. This observation follows from the fact that $R'_{i,j} \geq \frac{c_1}{m}R_{i,j}$ for every i and j. (This fact can be proved by integrating the values of $R_{i,j}$ and $R'_{i,j}$ over all vertical and horizontal cuts of the layout. Each cut will contain $r(m-r)$ pieces of edges of $R_{i,j}$ and $\frac{c_1}{m}r(m-r)$ pieces of edges of $R'_{i,j}$ where r and $m-r$ are the number of nodes on opposite sides of the cut.) Since $W_m(n) \geq R'$ and since $R' \geq 2^{-13}c_1\sqrt{\alpha}m^2n^3 \log n$, we can conclude that (for a sufficiently small constant α) $W_m(n) \geq \alpha m^2n^2 \log n$, thus proving the inductive hypothesis.

In the second case, we can show by a simple contradiction argument (just plug the claimed value back into the sum) that there exists an i such that

$$L_i \geq \frac{2^{-13}\sqrt{\alpha}m^2n^2 \log n 2^i}{\beta i^2}$$

where β is the constant $\sum_{i=1}^{\infty} 1/i^2$. Using the straightforward relation

$$W_m(n) \geq 2^{2i}W_m(n2^{-i}) + L_i,$$

we can conclude that

$$W_m(n) \geq 2^{2i}\alpha m^2(n2^{-i})^2(\log n - i)^2 + \frac{2^{-13}\sqrt{\alpha}m^2n^2 \log n\, 2^i}{\beta i^2}$$

$$\geq \alpha m^2n^2 \log^2 n - 2i\alpha m^2n^2 \log n + \frac{2^{-13}\sqrt{\alpha}m^2n^2 \log n\, 2^i}{\beta i^2}$$

which is at least $\alpha m^2 n^2 \log^2 n$ for a sufficiently small constant α. This completes the proof of the area lower bound. We next prove the minimax wire length lower bound.

8.2.2. Wire Length Bound

From the proof of the wire area lower bound, we know that one of the following inequalities must hold:

$$R \geq 2^{-13}\sqrt{\alpha} m^3 n^3 \log n, \quad \text{or}$$

$$\sum_{i=1}^{\log n} L_i 2^{-i} \geq 2^{-13}\sqrt{\alpha} m^2 n^2 \log n.$$

When the first inequality holds, we showed that $W_m(n) \geq \Omega(m^2 n^3 \log n)$. Since $Q_{m,n}$ has $\Theta(mn^2)$ edges, this means that at least one of the edges in the layout must have length $\Omega(mn \log n) \geq \Omega(mn \log n / \log\log n)$. When the second inequality holds, a simple contradiction argument (as before, just plug the values back into the sum) can be used to show that either

1) there is an $i \leq 6 \log\log n$ such that $L_i \geq \Omega(m^2 n^2 \log n \, 2^i / \log\log n)$, or

2) there is an $i > 6 \log\log n$ such that $L_i \geq \Omega(m^2 n^2 \log n \, 2^i / i^2)$.

Since there are $mn2^{i+1}$ level i edges in $Q_{m,n}$, the first condition insures that the layout contains a wire of length $\Omega(mn \log n / \log\log n)$. The analysis of the second case is somewhat more difficult.

Consider a layout for $Q_{m,n}$ which achieves the minimax edge length and (among layouts which satisfy this constraint) has minimum area. Since $W_m(n) \geq L_i$ for all i, the second inequality implies that

$$W_m(n) \geq \Omega(m^2 n^2 \log^7 n / \log\log^2 n)$$

$$\geq \Omega(m^2 n^2 \log^6 n)$$

for this layout. Thus (without loss of generality) the horizontal length of the layout is at least $\Omega(mn \log^3 n)$.

Partition the layout into three equal-area vertical strips. By the minimality of the layout area, we can conclude that each of the outer strips contains $\Omega(mn \log^3 n)$ nodes. (Otherwise, a smaller layout with identical minimax edge length could be constructed.)

Since each mesh of trees has diameter $O(\log n)$, each mesh of trees must be entirely contained in an $O(mn \log^2 n)$ by $O(mn \log^2 n)$ rectangle. (Otherwise, there would be an edge of length $\Omega(mn \log n)$ and we would be done.) Thus nodes in the same mesh of trees must be grouped together in the layout. Since each mesh of trees contains $\Theta(n^2)$ nodes, the outer strips must contain $\Omega(\frac{m \log^3 n}{n})$ complete meshes of trees. Thus at least $\Omega(\frac{m \log^3 n}{n}) \geq \Omega(\frac{m}{n})$ nodes of each expander graph are contained in the left and right outer strips of the layout. Since any two sets of r_1 and r_2 nodes are linked by a path of length $O(\log \frac{m}{r_1} + \log \frac{m}{r_2})$ in an m-node expander graph, this means that there is a path of length $O(\log n)$ connecting the left outer strip to the right outer strip. As the strips are separated by a distance $\Omega(mn \log^3 n)$, we can conclude that the layout contains an edge of length $\Omega(mn \log^2 n)$. This completes the proof of the minimax wire length lower bound. We next prove the crossing number lower bound.

8.2.3. Crossing Number Bound

Let $C_m(n)$ denote the minimum crossing number of $Q_{m,n}$. As was the case with the wire area lower bound, we will show by induction on n that

$$C_m(n) \geq \alpha m^2 n^2 \log n$$

for a sufficiently small (but positive) constant α. The basis of the induction follows from the fact that $C \geq \Omega(B^2)$ for any N-node graph with bisection width $B >> \Omega(\sqrt{N})$. This fact immediately implies that the crossing number of an m-node expander graph is $\Omega(m^2)$. In what follows, we will assume that the hypothesis is true for all values less than n in order to prove it for n.

Consider a drawing of $Q_{m,n}$ in the plane which has $C_m(n)$ crossings. By the optimality of $C_m(n)$, we can assume that no pair of edges cross more than once and that pairs of edges incident to the same node do not cross at all. Using the drawing for $Q_{m,n}$, construct a drawing for a graph with $\Omega(m^2n^4)$ edges and mn^2 nodes as follows.

1. Draw an edge between every pair of nodes in the same expander graph which are incident to crossing expander graph edges.

2. Draw an edge between pairs of leaves in the same mesh of trees.

3. Draw an edge between pairs of leaves separated by a path of length 1 or 2 in the graph formed by steps 1 and 2 above.

4. Eliminate multiple edges.

Each edge in the new graph should be drawn along the edges of $Q_{m,n}$ in the natural way (e.g., the edges introduced in step 1 are drawn along the corresponding crossing edges of $Q_{m,n}$). It is not difficult to check that each expander edge is traced over at most m times during step 1 and that each level i mesh of trees edge is traced over at most $n^3 2^{-i}$ times in step 2. These values are multiplied by a factor of $O(n^2)$ for expander edges and $O(m)$ for mesh of trees edges by step 3.

Since every drawing of an m-node expander graph has $\Omega(m^2)$ crossings, it is not difficult to see that the resulting graph (even after step 4) has $E = \Omega(m^2n^4)$ edges and $N = mn^2$ nodes. In Theorem 7-6 of [19], Leighton shows that any drawing of such a graph must have $\Omega(E^3/N^2) = \Omega(m^4n^8)$ crossings. Thus

$$sm^2n^4 + \sum_{i=1}^{\log n} r_i m^2 n^5 2^{-i} + \sum_{i,j=1}^{\log n} t_{i,j} m^2 n^6 2^{-i-j} \geq \Omega(m^4n^8)$$

where $t_{i,j}$ is the number of crossings in the drawing of $Q_{m,n}$ involving a level i edge and a level j edge, r_i is the number of crossings involving a level i edge and an expander edge, and s is the number of crossings involving two expander edges. This means that one of the following inequalities must be true:

$$s \geq \Omega(m^2n^4),$$

$$\sum_{i=1}^{\log n} r_i 2^{-i} \geq \Omega(m^2 n^3)$$

or

$$\sum_{i,j=1}^{\log n} t_{i,j} 2^{-i-j} \geq \Omega(m^2 n^2).$$

If the first inequality holds, then we can conclude that

$$C_m(n) \geq s \geq \Omega(m^2 n^4) \geq \alpha m^2 n^2 \log n$$

for sufficiently small α. If the second inequality holds, then

$$C_m(n) \geq \sum_{i=1}^{\log n} r_i \geq \sum_{i=1}^{\log n} r_i 2^{-i} \geq \Omega(m^2 n^3) \geq \alpha m^2 n^2 \log n$$

for sufficiently small α. The analysis for the third case is somewhat more difficult.

Let $t_i = \sum_{j=i}^{\log n} t_{i,j}$ be the number of crossings involving a level i edge and a level j edge where $j \geq i$. When the third inequality holds, it is clear that $\sum_{i=1}^{\log n} t_i 2^{-2i} \geq \Omega(m^2 n^2)$. Thus there is an i such that $t_i \geq \Omega(m^2 n^2 2^i)$. Using the inductive hypothesis, we can thus conclude that

$$\begin{aligned} C_m(n) &\geq 2^{2i} C_m(n2^{-i}) + t_i \\ &\geq 2^{2i} \alpha m^2 (n2^{-i})^2 (\log n - i) + t_i \\ &= \alpha m^2 n^2 \log n - i\alpha m^2 n^2 + \Omega(m^2 n^2 2^i) \end{aligned}$$

which is at least $\alpha m^2 n^2 \log n$ for a sufficiently small constant α. This concludes the proof of the crossing number lower bound and of the theorem. ∎

9. Remarks

The divide-and-conquer strategy based on bifurcators has also been successfully applied to the study of three-dimensional VLSI layouts [23]. In addition, the techniques and results are applicable to graph and data-structure embeddings, and also provide bounds on one- and two-dimensional bandwidth minimization.

There are a number of problems left unresolved in this paper. Some of the more important ones are mentioned below.

1. How much area is needed to lay out an N-node planar graph? The best universal upper bound is $O(N \log^2 N)$ [26, 45] while the best existential lower bound (for the tree of meshes) is $\Omega(N \log N)$ [19, 20].

2. Is there a polynomial time algorithm for laying out trees with edges not much longer than the minimax edge length? The best tree layout algorithm known produces layouts with edges of

length $\Theta(\sqrt{N}/\log N)$[3]. Although this is optimal for some trees, it is way off for others. On the other hand, it is NP-Complete to determine if a tree can be laid out with all edges of length one [2].

3. Is there a better way to realize a network in an environment that contains defective processors? Theorem 15 guarantees that any graph can be realized using the good processors provided the "channels" have width $\Omega(\frac{F}{\sqrt{N}} \log \frac{N}{F})$ in a regular layout. This bound is clearly optimal for some networks (such as expander-connected meshes of trees) but is not known to be optimal for simpler networks. In particular, it is not known whether or not a constant number of tracks per channel suffices to configure a mesh from the good processors. Since $F = \sqrt{N}$ for an N-node mesh, the best known upper bound on channel width is $O(\log N)$.

4. Is there a provably good, polynomial time algorithm for the bisection width problem? Although the bisection width problem is known to be NP-complete [13], there are many heuristics which do quite well in practice [6, 7, 10, 18, 37, 40]. Analyzing these or developing new heuristics along similar lines may help solve the layout problem.

5. Is there a provably good, polynomial time algorithm for the crossing number problem? This problem was recently shown to be NP-complete [14], but the possibility of approximation algorithms is not ruled out. The arguments of Section 7 suggest that graph bisection algorithms might be effective for this problem.

Acknowledgements

Charles Leiserson was a major collaborator in this research. Most of the problems, and their solutions, came up in discussions with him. Thanks too, for helping us through endless versions. We are grateful to Tom Lengauer, Kurt Mehlhorn, Gary Miller, Franco Preparata, Ron Rivest, Arnie Rosenberg, Jim Saxe, Larry Snyder, and Clark Thompson for their helpful remarks. Finally, special thanks to the referee for putting up with an error-filled earlier draft and for providing detailed and insightful comments.

References

[1] J. Bentley and H. T. Kung, "A tree machine for searching problems," *Proceedings of the 1979 International Conference on Parallel Processing, IEEE* (1979).

[2] S. N. Bhatt and S. Cosmadakis, "The complexity of minimizing wire lengths in VLSI Layouts," unpublished manuscript, M.I.T., (1982).

[3] S. N. Bhatt and C. E. Leiserson, "Minimizing the longest edge in a VLSI layout," M.I.T. VLSI Memo 82-86, (1982).

[4] S. N. Bhatt and C. E. Leiserson, "How to assemble tree machines," *Fourteenth Annual ACM Symposium on Theory of Computing* (1982).

[5] G. Bilardi, M. Pracchi, and F. Preparata, "A critique and appraisal of VLSI models of computation," *Proceedings CMU Conference on VLSI Systems and Computations* (1981).

[6] M. A. Breuer, "Min-cut placement," *Journal of Design Automation and Fault Tolerant Computing* Vol. 1, No. 4, (October 1977), 343-362.

[7] T. Bui, *On Bisecting Random Graphs,* S.M. thesis, Dept. of Electrical Engineering and Computer Science, (1983). Also appears as M.I.T. LCS Technical Report 287.

[8] P. R. Cappello and K. Steiglitz, "Area-efficient VLSI structures for multiplying at clock rate," Technical Report 289, Department of EECS, Princeton University, (1981).

[9] D. Dolev, F. T. Leighton, and H. Trickey, "Planar embeddings of planar graphs," M.I.T LCS Technical Memo 237, (1983).

[10] C. M. Fiduccia and R. M. Mattheyses, "An almost linear algorithm for partitioning networks," unpublished manuscript, (1982).

[11] R. Floyd and J. Ullman, "The compilation of regular expressions into integrated circuits," *Twenty-First Annual IEEE Symposium on Foundations of Computer Science* (1980).

[12] O. Gabber and Z. Galil, "Explicit constructions of linear size superconcentrators," *Proceedings 20th Annual IEEE Symposium on Foundations of Computer Science* (1979), 364-370.

[13] M. R. Garey and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness,* Freeman and Company, San Francisco, (1979).

[14] M. R. Garey and D. S. Johnson, "unpublished manuscript," (1982).

[15] J. R. Gilbert, *Graph Separator Theorems and Sparse Gaussian Elimination,* Ph.D. thesis, Dept. of Computer Science, Stanford University, (1980).

[16] C. Goldberg and D. West, "Even splittings of circle colorings," unpublished manuscript, (1982).

[17] J. Greene and A. El Gamal, "Area and delay penalties in restructurable wafer-scale arrays," *Third Caltech Conference on VLSI* R. Bryant, ed., Computer Science Press, (1983).

[18] B. W. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs," *The Bell System Technical Journal* (1970), 291-307.

[19] F. T. Leighton, *Layouts for the Shuffle-Exchange Graph and Lower Bound Techniques for VLSI,* Ph.D. thesis, Dept. of Mathematics, Massachusetts Institute of Technology, (1981). A revised version appears as *Complexity Issues in VLSI,* Foundations of Computing Series, M.I.T. Press (1983).

[20] F. T. Leighton, "New lower bound techniques for VLSI," *Twenty-Second Annual Symposium on Foundations of Computer Science, IEEE* (1981).

[21] F. T. Leighton, "A layout strategy for VLSI which is provably good," *Fourteenth Annual ACM Symposium on Theory of Computing* (1982).

[22] F. T. Leighton and C. E. Leiserson, "Wafer-scale integration of systolic arrays," *Twenty-Third Annual IEEE Symposium on Foundations of Computer Science* (1982).

[23] F. T. Leighton and A. L. Rosenberg, "Three dimensional circuit layouts," M.I.T. VLSI Memo 102, (1982).

[24] C. E. Leiserson, "A model for VLSI computation," Thesis proposal, CMU, (1979).

[25] C. E. Leiserson, "Systolic priority queues," *Proceedings of the Caltech Conference on Very Large Scale Integration,* C. Seitz, ed., California Institute of Technology, (1979).

[26] C. E. Leiserson, "Area-efficient layouts (for VLSI)," *Twenty-First Annual Symposium on Foundations of Computer Science, IEEE* (1980).

[27] C. E. Leiserson, *Area-Efficient VLSI Computation,* Ph.D. thesis, Dept. of Computer Science, Carnegie-Mellon University, (1981). Also published by M.I.T. Press 1983.

[28] 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* (1965).

[29] R. J. Lipton and R. E. Tarjan, "A separator theorem for planar graphs," *A Conference on Theoretical Computer Science,* University of Waterloo, (1977).

[30] G. A. Magó, "A network of microprocessors to execute reduction languages, Parts I and II," *International Journal of Computer and Information Sciences* (December, 1979).

[31] C. Mead and L. Conway, *Introduction to VLSI Systems,* Addison-Wesley, (1980).

[32] K. Mehlhorn, "personal communication," (1982).

[33] D. Nath, S. N. Maheshwari, and P. C. P. Bhatt, "Efficient VLSI networks for parallel processing based on orthogonal trees," *IEEE Transactions on Computers* (July 1983).

[34] M. Paterson, W. Ruzzo, and L. Snyder, "Bounds on minimax edge length for complete binary trees," *Thirteenth Annual ACM Symposium on Theory of Computing* (1981).

[35] J. Raffel, "On the use of nonvolatile programmable links for restructurable VLSI," *Proceedings of the Caltech Conference on Very Large Scale Integration* (1979).

[36] V. Ramachandran, "On driving many long lines in a VLSI layout," *Proceedings Twenty third Annual IEEE Symposium on Foundations of Computer Science* (1982).

[37] R. L. Rivest, "The "PI" (placement and interconnect) system," *Proceedings 19th Annual Design Automation Conference* (1982).

[38] A. Rosenberg, "Routing with permuters: toward reconfigurable and fault-tolerant networks," Technical Report CS-1981-13, Duke University, (1981).

[39] W. Ruzzo and L. Snyder, "Minimum edge length planar embeddings of trees," *Proceedings CMU Conference on VLSI Systems and Computation* (1981).

[40] A. Sangiovanni-Vincentelli, L. Chen, and L. Chua, "An efficient heuristic cluster algorithm for tearing large-scale networks," *IEEE Transactions on Circuits and Systems* Vol. CAS-24, No. 12, (1977), 709-717.

[41] C. D. Thompson, "Area-time complexity for VLSI," *Eleventh Annual ACM Symposium on Theory of Computing* (1979).

[42] C. D. Thompson, *A Complexity Theory for VLSI,* Ph.D. thesis, Dept. of Computer Science, Carnegie-Mellon University, (1980).

[43] J. D. Ullman, *Computational Aspects of VLSI,* Computer Science Press, (1983).

[44] L. G. Valiant, "On non-linear lower bounds in computational complexity," *Proceedings Seventh Annual Symposium on Theory of Computing* (1975).

[45] L. G. Valiant, "Universality considerations in VLSI circuits," *IEEE Transactions on Computers* (February, 1981).

CS-TR Scanning Project
Document Control Form

Date : 7/14/95

Report # LCS-TR-305

Each of the following should be identified by a checkmark:
Originating Department:

☐ Artificial Intellegence Laboratory (AI)
☒ Laboratory for Computer Science (LCS)

Document Type:

☒ Technical Report (TR) ☐ Technical Memo (TM)
☐ Other:_____

Document Information

Number of pages: 44(53-IMAGES)
Not to include DOD forms, printer instructions, etc... original pages only.

Originals are:
☒ Single-sided or
☐ Double-sided

Intended to be printed as :
☐ Single-sided or
☒ Double-sided

Print type:
☐ Typewriter ☐ Offset Press ☐ Laser Print
☐ InkJet Printer ☐ Unknown ☐ Other:_____

Check each if included with document:

☒ DOD Form(2) ☐ Funding Agent Form ☒ Cover Page
☒ Spine ☒ Printers Notes ☐ Photo negatives
☐ Other: _____

Page Data:

Blank Pages(by page number):_____

Photographs/Tonal Material (by page number):_____

Other (note description/page number):

Description :	Page Number:
IMAGE MAP: (1-44) PAGES #'ED	1-44
(45-50) SCANCONTROL, COVER, SPINE, PRINTER'S NOTES, DOD(2)	
(51-53) TRGT'S (3)	

Scanning Agent Signoff:
Date Received: 7/14/95 Date Scanned: 7/18/95 Date Returned: 7/20/95

Scanning Agent Signature: Michael W. Cook

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER MIT/LCS/TR-305	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) A Framework for Solving VLSI Graph Layout Problems		5. TYPE OF REPORT & PERIOD COVERED Interim Research 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Sandeep N. Bhatt and Frank Thomson Leighton		8. CONTRACT OR GRANT NUMBER(s) Contract # N00014-80-C-0622 DARPA/DOD
9. PERFORMING ORGANIZATION NAME AND ADDRESS MIT Laboratory for Computer Science 545 Technology Square Cambridge, MA 02139		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS DARPA/DOD/IPTO 1400 Wilson Boulevard Arlington, VA 22209		12. REPORT DATE September 1983 13. NUMBER OF PAGES 44
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Office of Naval Research/Dept of Navy Information Systems Program Arlington, VA 22209		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for Public Release; distribution is unlimited.

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

Unlimited

18. SUPPLEMENTARY NOTES

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

Bifurcator, bisection width, capacitive delay, configurable layout, crossing number, decomposition tree, graph layout, layout area, mesh of trees, propagation delay, separator theorem, tree of meshes, Very Large Scale Integration (VLSI), wafer-scale integration, wire length.

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

This paper introduces a new divide-and conquer framework for VLSI graph layout. Universally close upper and lower bounds are obtained for important cost functions such as layout area and propagation delay. The framework is also effectively used to design regular and configurable layouts, to assemble large networks of processors using restructurable chips, and to configure networks around faulty processors. It is also shown how good graph partitioning heuristics may be used to develop a probably good layout strategy.

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

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Scanning Agent Identification Target

Scanning of this document was supported in part by the **Corporation for National Research Initiatives,** using funds from the **Advanced Research Projects Agency** of the **United states Government** under Grant: **MDA972-92-J1029.**

The scanning agent for this project was the **Document Services** department of the **M.I.T Libraries.** Technical support for this project was also provided by the **M.I.T. Laboratory for Computer Sciences.**

Scanned

Date: 7/19/1995

M.I.T. Libraries
Document Services