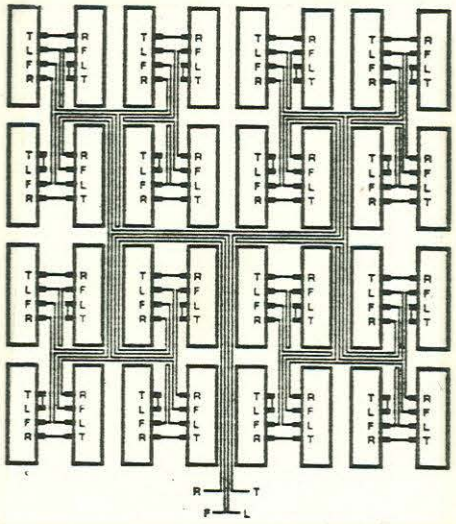
MIT/LCS/TM-255

HOW TO ASSEMBLE TREE MACHINES

Sandeep N. Bhatt

Charles E. Leiserson



March 1984

How to Assemble Tree Machines

Sandeep N. Bhatt
Charles E. Leiserson

Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

Abstract—Many researchers have proposed that ensembles of processing elements be organized as trees. This paper explores how large tree machines can be assembled efficiently from smaller components. A principal constraint considered is the limited number of external connections from an integrated circuit chip. We also explore the emerging capability of restructurable VLSI which allows a chip to be customized after fabrication.

We give a linear-area chip of m processors and only four off-chip connections which can be used as the sole building block to construct an arbitrarily large complete binary tree. We also present a restructurable linear-area layout of m processors with $O(\lg m)$ pins that can realize an arbitrary binary tree of any size. This layout is based on a solution to the graph-theoretic problem: Given a tree in which each vertex is either black or white, determine how many edges need be cut in order to bisect the tree into equal-size components, each containing exactly half the black and half the white vertices.

These ideas extend to more general graphs using separator theorems or bifurcators.

This research was supported by DARPA Grant N00014-80-C-0622.

1. Introduction

A tree may not be the best multiprocessor organization, but it has been proposed by many researchers for a variety of reasons. For example, a complete binary tree of processing elements can be the major component of a priority queue resource [15] and of a smart-memory raster graphics system [10]. A complete binary tree can also serve as a hardware structure for searching [2], for databases [29], or for direct execution of applicative programming languages [21]. Browning [6] proposes a complete binary tree for general-purpose multiprocessing.

Attention is also directed to binary trees which are not complete. Floyd and Ullman [8] show that strings described by a regular expression can be recognized by processing elements organized as the parse tree of the regular expression. Foster and Kung [9] have a similar scheme based on the simple configurable layout of Section 3 (first presented in [17]). There are other proposals, for example [27], of machine organizations which, though not trees, are nevertheless tree-like.

We shall not debate the merits of the various tree machine architectures here, but shall confine ourselves to understanding their physical organization. In this regard, one attraction of trees is that they can be laid out efficiently. Figure 1 shows the familiar H-tree layout originally proposed by Mead and Rem [22]. This layout of a complete binary tree requires linear area, as opposed to the $O(n \lg n)$ area standard layout shown in Figure 2. Leiserson [16] and Valiant [30] independently discovered that arbitrary binary trees could be laid out in linear area. In fact, Valiant proved that no crossovers were necessary in a linear-area layout. Based on ideas from Paterson, Ruzzo, and Snyder [23] and Bhatt and Leiserson [4], planar embeddings of arbitrary trees that minimize the maximum edge length were given by Ruzzo and Snyder [26].

Heretofore, the theoretical work on layouts has assumed that the entire tree fits on a chip. But the tree machines discussed above might be much larger. Whenever any system is larger than a single chip, it becomes necessary to partition it among separate chips which can be assembled at the circuit board (or chip carrier) level. What is the most effective way to partition a large tree among chips?

This question is pressing because although integrated circuit technology has been advancing at a breathtaking pace, one sector of that technology has been crawling in comparison. The technology for packaging chips severely limits the number of external connections to an integrated circuit, and whereas some enthusiastic technologists project an eye-opening 10^8 components per chip, two hundred pins per chip seems a large number to most. A chip that requires many more is unlikely to be realizable for quite some time.

Most of the theoretical work on tree layout has also implicitly assumed that a given tree, after masks have been made of the layout, will be replicated many times. This assumption is implicit because of the economics of integrated circuit fabrication technology: it is expensive to make one chip, but cheap to make many copies. For this economic reason, manufacturers of custom chips have been encouraged to make *configurable* designs such as gate-arrays, ROM's, and PLA's. The entire chip is manufactured except for one mask. The customer to whom the chip will be sold specifies a configuration of the chip, and the final layer of metalization connects up the circuitry in that particular way. Thus most of the design and fabrication costs are factored over many custom chips. Nevertheless, many copies must be made of the same custom chip for it to be economical.

Restructurable integrated circuits provide a means for the interconnections on a chip to be

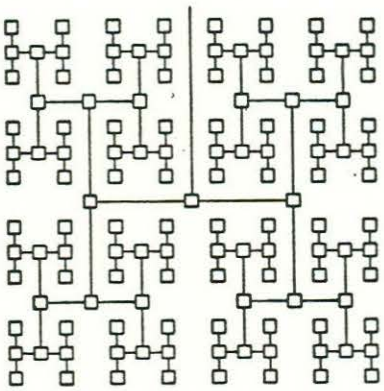


Figure 1: *The linear area "H-tree" layout of a complete binary tree.*

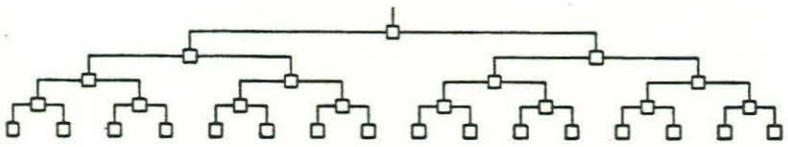


Figure 2: *An $O(n \lg n)$ layout of a complete binary tree.*

configured after fabrication. The most common example is a PROM (programmable read-only memory) in which diodes, which normally pass current, can be busted so that a connection is no longer made. More recent and exciting is the work on restructurable VLSI at IBM [20] and MIT Lincoln Laboratory [24]. Connections between two metal layers are produced reliably and efficiently by laser welding. Connections can also be broken by using the laser to cut wires in the circuit. Figure 3 shows a scanning electron microscope photograph of laser welds and cuts on a chip at MIT Lincoln Laboratory.

Restructurable VLSI chips have the advantage that the cost of quantity-of-one designs can still be factored over many chips, but some propose systems that included dynamically restructurable interconnections. For example, the proposed CHiP project at Purdue (Snyder [28]) is a dynami-

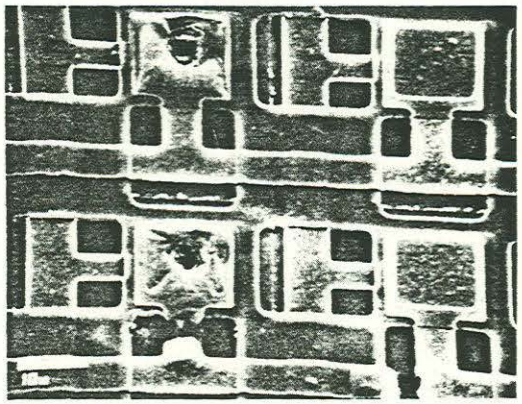


Figure 3: *Laser welds and cuts on a restructurable integrated circuit chip (courtesy of MIT Lincoln Laboratory).*

cally restructurable multiprocessor. It has not yet been demonstrated that large scale dynamically restructurable interconnections are economically feasible due to overheads in reliability, area, performance, and fabrication sophistication, but our results do indeed apply to dynamically restructurable layouts.

The rest of this paper addresses packaging constraints and restructurable VLSI with regard to tree layouts. Section 2 gives a chip with four pins that can be used as the sole building block for arbitrarily large, complete binary trees. A simple, but nonoptimal, restructurable layout that can implement any binary tree is given in Section 3. Section 4 proves a *two-color bisector theorem* for trees which is the main technical tool for producing the restructurable chip given in Section 5. This chip of M vertices has linear area and $O(\lg M)$ pins, and it can be used in quantity to assemble any binary tree of any size. Section 6 contains extensions and conclusions.

2. Packaging a complete binary tree

This section studies the problem of packaging complete binary trees, and presents the design of a single chip with four pins that can be used to bulid arbitrarily large complete binary trees. This chip, originally proposed in [17], has since been used (at the circuit board level) in tree-machine projects at Caltech and Bell laboratories [7].

We begin, however, by examining the inefficient partitioning of a complete binary tree proposed in [15] and elsewhere (for example, [6]). Each of the squares in Figure 4 is a *Type A* chip and is packed as full as possible with processors in the H-tree layout of Figure 1. The rectangle above is a *Type B* chip which contains the standard $O(n \log n)$ area layout of Figure 2, but with each leaf connected off-chip. The Type B chip can be used repeatedly to combine several smaller complete binary trees into a larger one.

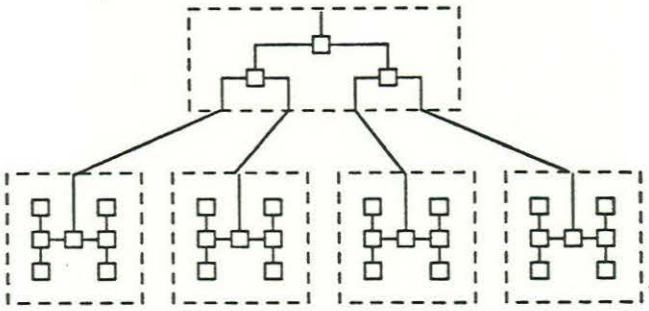


Figure 4: *An inefficient partitioning of a complete binary tree into Type A and Type B chips.*

Theorem 1. *Suppose Type A chips each contain $P = 2^p - 1$ vertices, and Type B chips each contain $Q = 2^q - 1$ vertices. Then a complete binary tree with at least $N = 2^n - 1$ vertices can be assembled from*

- $\frac{N+1}{P+1}$ *Type A chips and*
- $\left\lceil \frac{N-P}{Q(P+1)} \right\rceil$ *Type B chips.*

Proof. The complete binary tree can be assembled using the scheme from Figure 4.∎

We can do better, however. Figure 5 shows a *Type C* chip with only four off-chip connections. Arbitrarily large complete binary trees can be assembled from this one kind of chip. Each chip contains one internal node of the tree, and the remainder of the chip is packed as full as possible with an H-tree layout. The internal node requires three off-chip connections (denoted F, R, and L in the figure) for its father, right son, and left son. The H-tree requires only one off-chip connection (denoted T) to its father.

Theorem 2. *Suppose Type C chips each contain $M = 2^m$ vertices. Then a complete binary tree with at least $N = 2^n - 1$ vertices can be assembled from $(N+1)/M$ Type C chips.*

Proof. We show how arbitrarily large complete binary trees can be built up. To interconnect two chips, the unconnected internal node of one of the two chips is selected as the father of the two H-trees. In Figure 6 the internal node on the left has been chosen for this purpose. The R pin on this chip is connected to its own T pin, and the L pin is connected to the T pin on the other chip. Considered as a unit, the combined two chips now have the same structure as a single chip—three connections to an internal node and one to the root of a complete binary tree. The pair of chips can be similarly combined with another pair to produce a quadruple of chips, which can in turn be combined, and so forth inductively, as is shown in Figure 7.∎

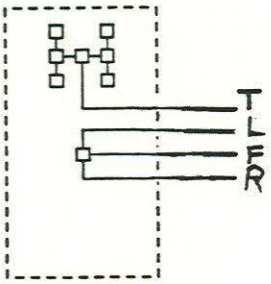


Figure 5: *Only one Type C chip is needed to package a complete binary tree.*

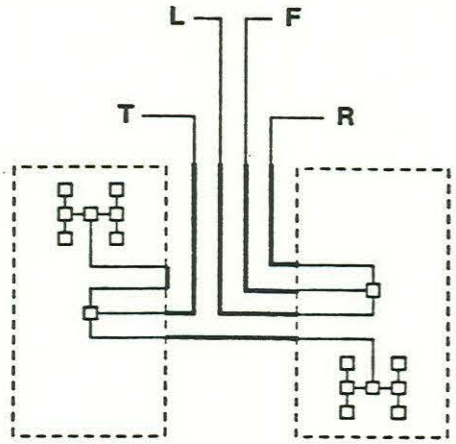


Figure 6: *Wiring two Type C chips together.*

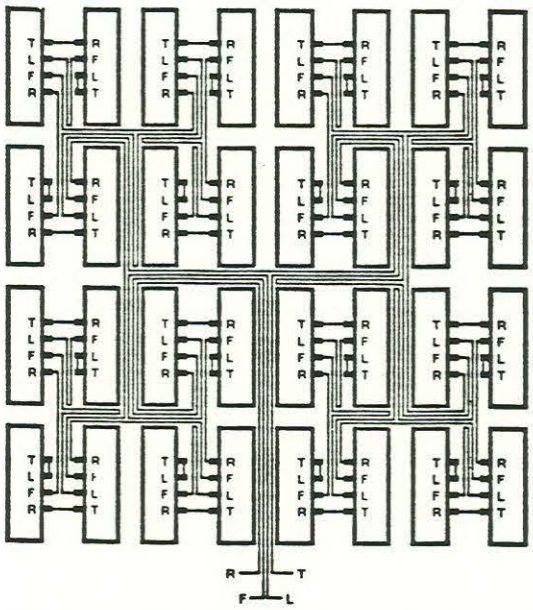


Figure 7: *A large complete binary tree assembled from many Type C chips.*

The one-chip method has many advantages over the two-chip method. Most obviously, the one-chip method uses only one kind of chip. Why manufacture two kinds when one will do? Second, only four data paths go off chip. Third, the Type C chip is packed full, while the Type B chip is almost empty because it is pin bound. Finally, the area of the assembly (on a circuit board for example) is linear in the number Type C chips used. The two-chip solution gives an $O(n \log n)$ area circuit layout. Although the case is not particularly strong for asymptotic analysis of circuit layout, the constant factors give a clear preference to the more regular, linear area layout. If circumstances permit, the wires connecting the chips can in fact be routed underneath the chips themselves, thereby requiring no more area on the circuit board than the chips themselves.

3. A restructurable chip for packaging arbitrary trees

This section presents a simple (but suboptimal) scheme for packaging arbitrary trees using a single restructurable chip. The solution is suggested by a technique of Bentley and Leiserson [17] for producing *collinear layouts* for arbitrary trees. The strategy for producing collinear layouts is, in turn, based on the observation that trees have a small *separator theorem.* This section defines separator theorems, describes the strategy for producing collinear layouts, and proposes a simple packaging scheme. Although the solution is asymtotically suboptimal, the results are crucial to the optimal scheme presented in the next section.

Separator theorems [19] have been applied to solve a variety of graph-theoretic problems including graph layout (for example, [3, 14, 16, 17, 30]). Formally, let Ψ be a family of graphs closed under the subgraph relation, and let $\alpha \leq 1/2$ and β be positive constants. If every graph on n vertices in Ψ can be separated into two disconnected components, each having at least $\lfloor \alpha n \rfloor$ vertices, by removing no more than $\beta f(n)$ edges, then Ψ has an $f(n)$-*separator theorem.*

By removing a single edge, any n-vertex binary tree can be separated into two components, each with no more than $\lfloor \frac{2}{3} N \rfloor + 1$ vertices [18]. (The worst-case occurs for the four-vertex tree in which one vertex is adjacent to three others.) Either of the two components may be a forest, but since the same result applies to forests, the binary tree can be split recursively. Since each of the recursively generated subgraphs can be split by removing a single edge, the class of binary trees has a *one-separator theorem.*

Bentley and Leiserson [16] used the one-separator theorem for trees to produce *collinear layouts* for binary trees. In a collinear layout all the vertices are placed along a common baseline, and tree edges are routed along horizontal and vertical tracks on one side of the baseline, as seen in Figure 8. The *height* of a collinear layout is defined as the number of distinct horizontal tracks used for routing the edges. As shown in the following theorem, efficient collinear layouts can be produced using the one-separator theorem for binary trees. (In fact, Yannakakis [31] has shown that a minimum height layout can be obtained for a given N-vertex tree in $O(N \lg N)$ time.)

Lemma 3. *Every N-vertex binary tree has a collinear layout with height no greater than* $\lg N$.

Proof. Using the one-separator theorem, first separate the tree. If either component contains more than $N/2$ vertices, separate it into two smaller components using the one-separator theorem again. Next, recursively construct collinear layouts for each subforest, and place these layouts

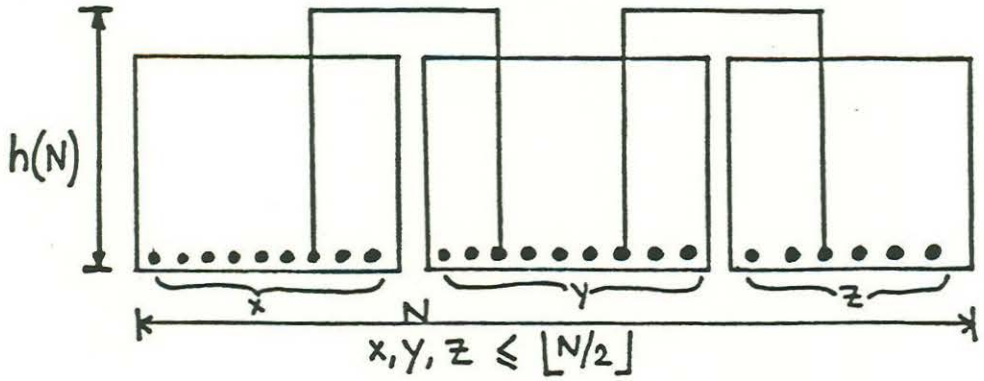


Figure 8: *The construction of a collinear layout.*

side-by-side along the baseline. Finally, as shown in Figure 8, connect the two (or three) subforests by routing the separator edges on distinct vertical tracks and along a common horizontal track. (For two components this is trivial since only edge is routed; for three components, place the subforest connected to both other subforests in the middle as shown.) For each node there are three vertical tracks to accomodate edges incident to that node.

The height of the layout is determined by a simple recurrence relation. Let $h(N)$ be the height of the layout, so that $h(1) = 0$, and in general,

$$h(N) \leq h(\lfloor N/2 \rfloor) + 1 .$$

A straightforward calculation yields $h(N) \leq \lg N$. ∎

Corollary 4. *Any binary tree with N vertices can be bisected into components of sizes $\lfloor N/2 \rfloor$ and $\lceil N/2 \rceil$ by removing at most $\lg N$ edges.*

Proof. Consider the vertical line that passes midway through the collinear layout. It bisects the N vertices and the number of edges it cuts is no more than $\lg N$, the height of the layout. ∎

The collinear layout can also be used to make a configurable chip of N vertices which can realize any N-vertex binary tree. The chip consists of N collinear vertices, with three vertical wires connected to each vertex, and $\lg N$ contacts along each vertical wire. Every N-vertex binary tree can be configured on this chip by specifying one extra custom layer. The custom layer consists of the portions of the wires in the collinear layout that run horizontally. The horizontal wires run between the rows of contacts, and spurs to the contacts make connections.

An unattractive feature of the configurable chip is that a different mask must be designed for each tree. Not surprisingly, the same idea can be used to design a *restructurable* chip for trees, where the chip is customized (for example, by laser) after fabrication. Once again, the collinear layout serves as the basis for the design. The restructurable chip consists of vertical wires running the height of the layout on one layer, and horizontal wires running the width of the layout on

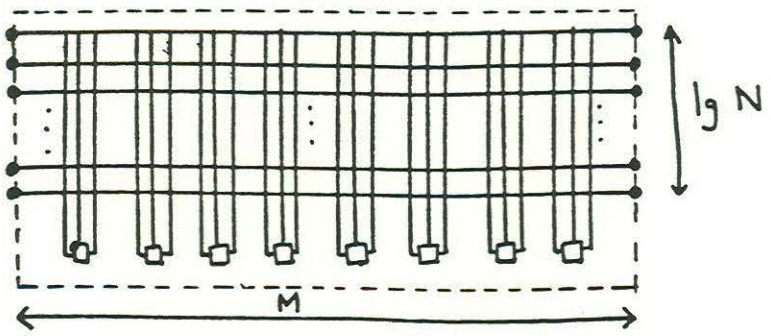


Figure 9: *A Type D restructurable chip which can be used to assemble large binary trees by making and breaking connections.*

another. By using laser welds to connect various horizontal wires to appropriate vertical wires, and laser trimming to break horizontal wires, any tree can be realized in accordance with its collinear layout. The number of connections made or broken is $O(N)$.

This restructurable layout also suggests a method of packaging arbitrary binary trees using a single *Type D* restructurable chip, which is shown in Figure 9. From each of the collinear vertices, three vertical wires are run. At every intersection of a horizontal and vertical wire is a weld point which can be programmed after fabrication. Each horizontal wire is connected to pins at either end.

Theorem 5. *Suppose Type D chips each contain M vertices and n horizontal wires. Then any binary tree with $N = 2^n$ vertices can be realized with $\lceil N/M \rceil$ Type D chips.*

Proof. Take the $\lceil N/M \rceil$ chips and place them side by side in the natural way hooking up adjacent pins. Following Lemma 3, draw a collinear layout of height at most $\lg N$ for the N-vertex tree. Map the layout onto the assembly in the obvious manner. Make and break connections on each chip to realize the layout.∎

Unfortunately, if a tree with more than 2^n vertices were required, this chip might not be able to configure it. In the next section a better packaging scheme is developed whereby one restructurable chip containing M vertices in linear area and $O(\lg M)$ pins, can be used to package arbitrarily large binary trees.

Some restructurable technologies do not allow connections to be broken, and thus the scheme of Theorem 5 will not work. A naïve alternative is to break every horizontal wire into M unit length segments. Each segment can be connected to vertical wires and to its neighboring segments on the same horizontal track. Unfortunately, programming the interconnect requires a large number of welds to be made on an edge connecting two vertices. The scheme from Theorem 5 requires only two welds for each edge.

Figure 10 shows a *Type E* restructurable chip which can realize any tree by making, but not breaking, connections such that only two welds are required per edge. The chip has $M = 2^m$

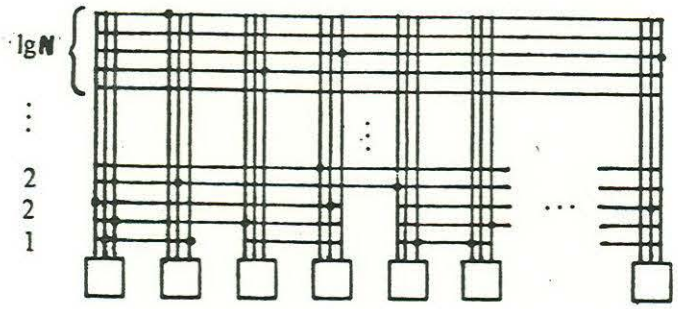


Figure 10: *A Type E restructurable chip which can be used to assemble large binary trees without breaking connections.*

vertices and n horizontal tracks which are divided into groups. The first group contains one horizontal track which consists of $M/2$ unit length wire segments. The second group contains two horizontal tracks, each with $M/4$ wire segments of length 2. In general, for $i = 1, 2, \ldots, m$, the ith group contains i tracks, each with $M/2^i$ wire segments of length 2^i. The remainder of the horizontal tracks are in group $m+1$. Each of these tracks has one wire of length M connected off chip.

Theorem 6. *Suppose Type E chips each contain $M = 2^m$ vertices and n horizontal tracks. Then any binary tree with $N = c2^{\sqrt{2n}}$ vertices can be realized with $\lceil N/M \rceil$ Type E chips, where c is a constant ($c \approx 1/\sqrt{2}$).*

Proof. Lay the $\lceil N/M \rceil$ chips side by side, and connect the pins to continue the on-chip grouping scheme such that for $i = 1, 2, \ldots, \lg N$, group i contains i tracks, each with $N/2^i$ wire segments of length 2^i. The total number of horizontal tracks is

$$\begin{aligned} h(N) &= 1 + 2 + \cdots + \lg N \\ &= \tfrac{1}{2} \lg N (\lg N + 1) \\ &\leq \tfrac{1}{2}\Big(\lg c + \sqrt{2n}\Big)\Big(\lg c + \sqrt{2n} + 1\Big) \\ &= n - \frac{1}{8}, \end{aligned}$$

for $c = 1/\sqrt{2}$, and thus n tracks are sufficient.

Observe that this assembly without its top group of $\lg N$ horizontal wires forms two smaller versions of itself. To realize a given tree, remove the $\lg N$ bisector edges as in Corollary 4, and recursively lay out the equal size components within the two smaller layouts. Combine the sublayouts by routing the bisector edges along the top group of wires that run across the layout. Since two connections are formed for each tree edge, the total number of welds is $2N - 2$.∎

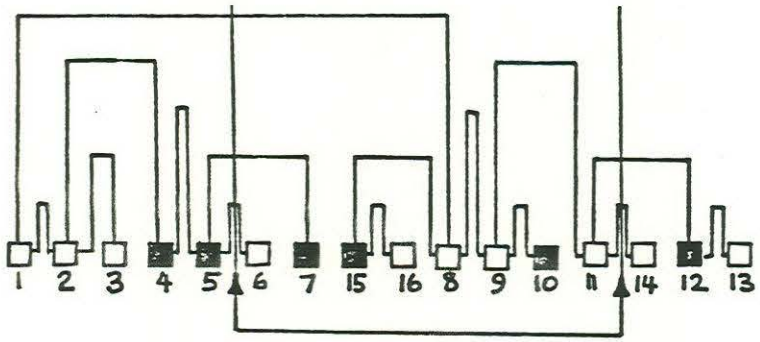


Figure 11: *At some point, a window of size $n/2$ slid along the base of the two-color collinear layout must contain half the white and half the black vertices.*

4. Two-color bisector theorems

Although the Type D restructurable chip with M vertices and $2n$ pin connections provides one way to package large trees, it suffers two disadvantages. First, it cannot be used to assemble trees with more than 2^n vertices. Second, and more important, the chip is wasteful in area. In fact, although every N-vertex tree can be laid out in $O(N)$ area [16, 30], a collinear layout for the complete binary tree requires at least $\Omega(N \lg N)$ area [5, 16]. Thus we are led to ask: *Does there exist a restructurable chip with M vertices, occupying $O(M)$ area, and having few pins which can realize every binary tree, no matter how large?*

In the next section we answer this question affirmatively. The question is fairly subtle, however, and does not follow as a straightforward application of the separator theorem. While we can effectively use the separator theorem to recursively bisect a tree into equal size components (as in Theorem 6), *there is nothing to bound the number of external edges that connect a component to the rest of the tree.* Thus for example, suppose we designed a chip with M vertices and P pins for packaging arbitrarily large trees. How can we guarantee that every tree can be decomposed into subgraphs of size at most M such that each component has no more than P external edges?

In this section we introduce the notion of *two-color bisector theorems* which can be used to recursively bisect a graph while also bounding the number of external edges into each component. Moreover, trees have small two-color bisector theorems, so that the number of external edges into a component is also small. These results use arguments from the previous section. In the next section, we apply two-color bisector theorems to design an optimal packaging scheme for binary trees.

Definition. Suppose that an N-vertex graph G has b black vertices and w white vertices. A two-color bisector for G is a set of edges whose removal bisects G into two subgraphs each of size at least $\lfloor N/2 \rfloor$, and such that each contains at least $\lfloor b/2 \rfloor$ black and $\lfloor w/2 \rfloor$ white vertices.

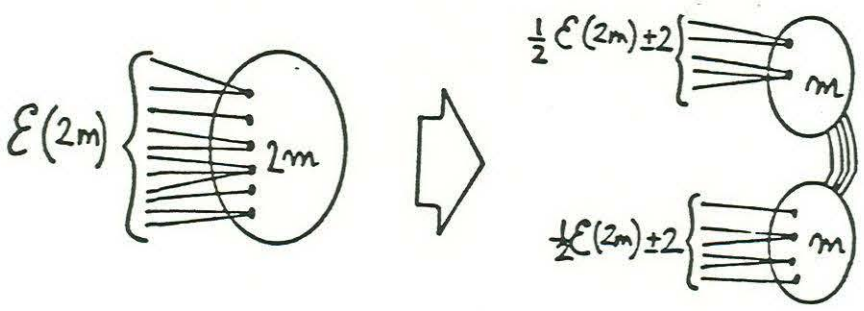


Figure 12: *To keep the number of external connections to all subcomponents small when a component is bisected, the external connections must be evenly divided between the subcomponents.*

Theorem 7. *Every N-vertex forest of binary trees has a two-color bisector of size $2\lg N$.*

Proof. Following Lemma 3, construct a collinear layout of height at most $\lg N$. Suppose there are b black vertices and $N-b$ white vertices. Consider a "window" which overlaps $\lfloor N/2 \rfloor$ consecutive vertices, and place it over the leftmost $\lfloor N/2 \rfloor$ vertices. If more than $\lceil b/2 \rceil$ black vertices fall within the window, slide the window one position to the right. Observe that by sliding the window one position, the number of black vertices within the window changes by at most one. Furthermore, by sliding the window all the way to the right, less than $\lfloor b/2 \rfloor$ black vertices would fall within the window. Consequently, there must be an intermediate placement of the window (see Figure 11) in which at least $\lfloor b/2 \rfloor$ black vertices and at least $\lfloor (N-b)/2 \rfloor$ white vertices are contained within the window. (Such a placement can be obtained in linear time.)

Draw vertical lines through the endpoints of the window in the position obtained above. The edges of the forest intersecting these lines form a two-color bisector of the forest. The size of this two-color bisector is no more than twice the height of the layout. Thus the size of the two-color bisector is no more than $2\lg N$.∎

For our purposes the following variant of two-color bisectors is more suitable. Suppose each vertex of an N-vertex forest is assigned a weight from a *bounded* set $\{1, 2, \ldots, k\}$ of weights. We wish to bisect the forest into two subforests, each of size at least $\lfloor N/2 \rfloor$, whose total weights differ by at most k. How many edges need be cut? Adapting the argument for two-color bisectors to this variant in a straightforward manner shows again that $2\lg N$ cuts suffice.

Having obtained bounds on the size of two-color bisectors for forests, we wish to use them for partitioning an arbitrarily large binary tree into subforests of size at most M so that every subforest has few edges connected to vertices in other subforests. This result is established in the following theorem.

Theorem 8. *Every N-vertex binary tree can be partitioned into $\lceil N/M \rceil$ subforests, each of size at most M, such that no subforest has more than $4\lg M + 8$ edges connected to vertices in other subforests.*

Proof. We prove the theorem for the case when $N = 2^i M$. The general case may be proved similarly, but we omit the tedious details of the analysis. As in Theorem 6, bisect the tree into two subforests, each of size at least $\lfloor N/2 \rfloor$, by cutting no more than $\lg N$ edges. Split each subforest recursively as follows. For each vertex in a recursively split component of size m assign a weight equal to the number of edges incident to that vertex and which were cut at a previous level. Since the degree of a vertex is at most three, the weight assigned to a vertex is at most 2. From the argument following Theorem 7, there is a weighted bisector of size no greater than $2 \lg m$ for the component. This weighted bisector divides the number of external connections almost equally (the difference is at most two) between the subcomponents of sizes $\lfloor m/2 \rfloor$ and $\lceil m/2 \rceil$. As seen in Figure 12, the number of external connections into either of the new subcomponents is no more than the size of the weighted bisector plus one-half the number of external connections into the component just split (plus two). This recursive decomposition terminates when each component has size at most M. Letting $\mathcal{E}(m)$ be the number of external connections into any component of size m, we have $\mathcal{E}(N) = 0$, and

$$\mathcal{E}(m) \leq \tfrac{1}{2}\mathcal{E}(2m) + 2\lg(2m) + 2 .$$

A little calculation shows that $\mathcal{E}(m) \leq 4 \lg m + 8$. This means that every subforest of size m in the recursive decomposition has at most $4 \lg m + 8$ external edges to other subforests. Substituting M for m, the result follows. ∎

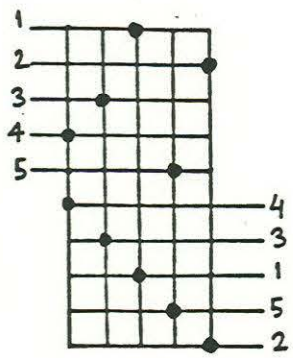


Figure 13: *A k-by-k restructurable permuter can realize any set of one-to-one connections between the terminals on the two sides.*

5. An optimal packaging scheme

The recursive decomposition of Theorem 8 leads directly to the design of an efficient restructurable chip which can be used in quantity to assemble any tree. This Type F restructurable chip has M vertices, $O(\lg M)$ pins, and an $O(M)$ area layout. This packaging scheme is the best possible when all vertices on the chips are utilized.

The design of the Type F chip uses *restructurable permuters*. A permuter P_k has k terminals on each side of a rectangle and can realize any one-to-one connection between the terminals. The switch shown in Figure 13 implements a permuter. It has dimensions $2k \times k$, with the terminals along the longer sides.

The construction of the Type F restructurable chip is recursive and follows the recursive decomposition of Theorem 8. We shall use R_m to denote a level of the recursive layout with m vertices, and let R_M denote the restructurable Type F chip of M vertices itself. Figure 14 shows how the Type F chip R_M is constructed from four copies of $R_{M/4}$, four copies of $P_{4\lg M}$, and two copies of $P_{4\lg M+4}$. Letting $S(M)$ be the length of the side of the layout, we have $S(1) = 1$ and,

$$S(M) \leq 2S(M/4) + O(\lg M)\,,$$

which yields $S(M) = O(\sqrt{M})$, so that the area is linear in M. The number of pins on R_M is $4\lg M + 8$. We now show that every large tree can be assembled using R_M.

Theorem 9. *Suppose Type F chips each contain M vertices. Then any N-vertex binary tree can be assembled using $\lceil N/M \rceil$ Type F chips, the minimum possible.*

Proof. As before, we assume that $N = 2^i M$, although the result extends in a straightforward manner to the general case. Following Theorem 8, decompose the tree into $\lceil N/M \rceil$ components, each of size at most M and having no more than $4\lg M + 8$ external edges to other components. Each of the $\lceil N/M \rceil$ components can be realized on a single Type F chip R_M. To see this, use

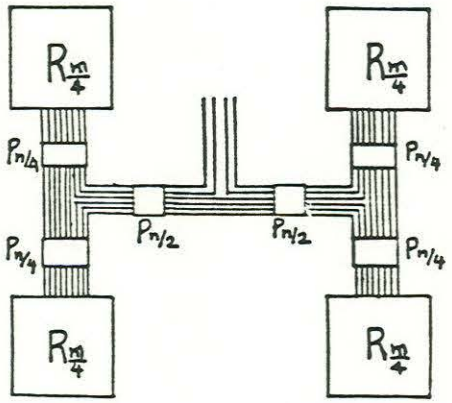


Figure 14: *The Type F restructurable chip R_M which can be used to assemble arbitrarily large binary trees.*

Theorem 8 to recursively decompose each component into single vertices. In this decomposition each subforest of size m has at most $4 \lg m + 8$ external edges. This decomposition may now be mapped directly onto the chip, using the permuters to route edges between different subcomponents. Since the number of external edges at any level is no greater than the size of the permuters at that level, the permuters can realize the desired routing. Vertices of the tree are embedded at fixed positions in the lowest level permuters P_1. Finally, each chip has enough pin connections so that the assembly can be completed off-chip by connecting the chips together as required by the original decomposition. (Permuters are not needed off chip because wires can be routed directly.)∎

The constant factors on area can be improved if one uses the smaller restructurable permuter P_k with dimensions $(k+O(\sqrt{k})) \times (k+O(\sqrt{k}))$ that follows from the channel routing algorithm of [1]. Whereas the simpler permuter from Figure 13 requires only two welds to make a connection, the more dense layout might require as many as k welds for each connection. Although the total number of welds required by either scheme is $O(M)$, the number per wire is $O(\lg M)$ if the simpler switch is used and $O(\lg^2 M)$ if the channel-routing permuter is used.

In related work, Rosenberg [25] has also considered permuters to obtain a degree of configurability in layouts.

6. Extensions and conclusions

All the layout techniques presented here extend to more general classes of graphs. In particular, the techniques extend to classes of graphs not closed under the subgraph relation by extending the definition of separator theorems as in 16 or 14 to apply recursively to graphs generated by the separator. For example, graphs with n^α-separator theorems have linear-area restructurable layouts if $\alpha < 1/2$. When $\alpha = 1/2$, the area is $O(n \lg^2 n)$, and if $\alpha > 1/2$, the area is $O(n^{2\alpha})$. These area bounds match the layout areas of 16 and 30 while requiring the layouts to

be restructurable. In each case the number of pins on a chip is $O(n^\alpha)$ if $\alpha > 0$, and $O(\lg n)$ if $\alpha = 0$.

These bounds are obtained by recursively using the separator theorem to produce a collinear layout and then chopping the layout with two cuts to yield a two-color bisector. There is one technical detail in using the extended notions of separator theorems in 16 and 14 to accomplish the cuts of the collinear layouts since we must make sure that the two-color bisector theorem applies recursively to the two halves of the graph. Rather than just cutting the edges incident to the two vertical lines, one must in addition cut a constant factor more edges in order that each of the subgraphs generated by the two-color bisector is the union of disjoint subgraphs generated by the separator theorem. A more general divide-and-conquer framework for this problem is given in [3].

The methods for tree assembly considered in this paper have all assumed that the overall utilization of the chips is 100 percent. specifically, only $\lceil N/M \rceil$ chips are used to assemble an N-vertex tree with chips that hold M vertices. Not surprisingly, if the assumption of full utilization is relaxed, fewer pins are needed. In particular, we can guarantee 50 percent utilization with six-pin chips using an idea due to Tom Leighton.

The assembly is generated recursively as in Section 5. At each step of the divide-and-conquer construction, there is a subforest A with at most six external connections. This subforest can always be split into two components, each containing at least one-sixth of the nodes and at most six external connections. We first use the standard separator theorem to remove one edge that splits A into two components B and C with at worst a $\frac{1}{3} : \frac{2}{3}$ ratio. The only case to worry about is if all the original external connections are incident to B (or to C) because the newly removed edge will now give B seven external connections. If this bad split indeed occurs, we split B further into B_1 and B_2 so that the seven connections are divided 3:4. (There is no constraint on the ratio of the size of B_1 to B_2.) Finally, we take whichever of B_1 and B_2 is smaller and combine it with C. Of the two remaining components, neither has more than six external connections, and each has at least $\lfloor |A| \rfloor / 6$ vertices.

The recursion terminates when any subforest has M or fewer vertices, in which case the subforest is embedded on a Type F chip. Of course, only six of the $O(\lg M)$ connections are actually used. The assembly method will never require more than $2\lceil N/(M+1) \rceil$ chips. The worst case occurs when every branch of the recursion terminates with the splitting of a subforest of size $M+1$. Higher utilization can be attained at the expense of more pins by generalizing this technique.

Since our discovery of two-color bisectors and their relation to restructurable layouts, they have been used in other VLSI layout problems. Based on partial knowledge of our work, Leighton 12 showed independently that any graph that has a $\sqrt{n}$-separator theorem can be embedded in his "tree of meshes," which is similar to the restructurable layout obtained when $f(n) = \sqrt{n}$. He and Rosenberg 13 have also used three-color bisector theorems to obtain optimal three-dimensional VLSI layouts.

The use of the collinear layout for obtaining a two-color bisector theorem from a separator theorem is combinatorially appealing, and can be recast as a *necklace* problem. Given a necklace of black and white pearls, how many cuts are necessary in order to divide the necklace into two pieces such that each of the pieces has the same (to within one) number of pearls of each

color? The obvious extension is to ask how many cuts are necessary to divide a necklace of k colors. Unfortunately, the naive idea of sliding a window across the collinear layout fails to work if $k \geq 3$. Recently, Goldberg and West 11 at Princeton, hearing of our open problem, developed an elegant topological argument to show that k cuts suffice, which is tight in that k cuts are necessary in some cases. This result implies, for example, that trees with k colors have $O(k \lg n)$ k-color bisectors and planar graphs with k colors have $O(k\sqrt{n})$ k-color bisectors.

Acknowledgments

Thanks to Tom Leighton of MIT for many helpful comments and his contributions to Section 6. Thanks to the RVLSI project group at MIT Lincoln Laboratory for Figure 3 and to Peter Schwarz of Carnegie-Mellon University for his help in preparing Figure 7.

References

[1] B. S. Baker, S. N. Bhatt, and F. T. Leighton, "An approximation algorithm for Manhattan routing," *Fifteenth Annual ACM Symposium on Theory of Computing* (1983).

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

[3] S. N. Bhatt and F. T. Leighton, "A framework for solving VLSI graph layout problems," *JCSS(to appear)* (1983).

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

[5] R. P. Brent and H. T. Kung, "On the area of binary tree layouts," *IPL* No. 11, (1980).

[6] S. Browning, *The Tree Machine: A Highly Concurrent Computing Environment,* Ph.D. thesis, Dept. of Computer Science, California Institute of Technology, (1980).

[7] S. Browning, "private communication," (April, 1981).

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

[9] M. Foster and H. T. Kung, "Recognize regular languages with programmable building-blocks," *VLSI 81,* J. Gray, ed., Academic Press, New York, (1981).

[10] H. Fuchs, J. Poulton, A. Paeth, and A. Bell, "Developing pixel-planes, a smart memory-based raster graphics system," *Proceedings, MIT Conference on Advanced Research in VLSI* P. Penfield, ed.,, (1982).

[11] C. Goldberg and D. West, "Bisection of circle colorings," (1983).

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

[13] F. Leighton and A. Rosenberg, "Three-dimensional circuit layouts," *Proceedings of the 1983 IEEE International Conference on Computer Design* (1983).

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

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

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

[17] C. Leiserson, *Area-Efficient VLSI Computation,* Ph.D. thesis, Dept. of Computer Science, Carnegie-Mellon University, (1981).

[18] P. Lewis, R. 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).

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

[20] J. Logue, W. Kleinfelder, P. Lowy, J. Moulic, and W. Wu, "Techniques for improving engineering productivity of VLSI designs," *Proceedings of the IEEE International Conference on Circuits and Computers* (1980).

[21] G. A. Magó, "A network of microprocessors to execute reduction languages, Parts 1 and 2," *International Journal of Computer and Information Sciences* (December, 1979).

[22] C. Mead and M. Rem, "Cost and performance of VLSI computing structures," *IEEE Journal of Solid State Circuits, Vol.SC-14, No. 2* (1979).

[23] 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).

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

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

[26] W. Ruzzo and L. Snyder, "Minimum edge length planar embeddings of trees," *VLSI Systems and Computations,* H. T. Kung, R. Sproull, and G. Steele, eds., Computer Science Press, (1981).

[27] C. Séquin, A. Despain, and D. Patterson, "Communication in X-tree, a modular multiprocessor system," *ACM 78 Proceedings* (1978).

[28] L. Snyder, "Overview of the CHiP computer," *VLSI 81,* J. Gray, ed., Academic Press, New York, (1981).

[29] S. Song, "A highly concurrent tree machine for database applications," *1980 International Conference on Parallel Processing* (1980).

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

[31] M. Yannakakis, "A polynomial algorithm for the min cut linear arrangement of trees," *Twenty Fourth Annual IEEE Symposium on Foundations of Computer Science* (1983).