LABORATORY FOR COMPUTER SCIENCE

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

MIT/LCS/TM-402

VLSI THEORY AND PARALLEL SUPERCOMPUTING

Charles E. Leiserson

May 1989

545 TECHNOLOGY SQUARE, CAMBRIDGE, MASSACHUSETTS 02139

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) MIT/LCS/TM-402	5. MONITORING ORGANIZATION REPORT NUMBER(S) N00014-87-K-825
6a. NAME OF PERFORMING ORGANIZATION MIT Laboratory for Computer Science 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION Office of Naval Research/Department of Navy
6c. ADDRESS (City, State, and ZIP Code) 545 Technology Square Cambridge, MA 02139	7b. ADDRESS (City, State, and ZIP Code) Information Systems Program Arlington, VA 22217
8a. NAME OF FUNDING/SPONSORING ORGANIZATION DARPA/DOD 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code) 1400 Wilson Boulevard Arlington, VA 22217	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO / WORK UNIT ACCESSION NO

11. TITLE (Include Security Classification)
VLSI Theory and Parallel Supercomputing

12. PERSONAL AUTHOR(S)
Leiserson, Charles E.

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Technical	FROM _____ TO _____	1989 May	14

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Fat-trees, hypercubes, integrated circuits; interconnection networks; layout theory, parallel computing, supercomputing; universality; Thompson's model; tree-of-meshes.

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

Since its inception, VLSI theory has expanded in many fruitful and interesting directions. One major branch is layout theory which studies the efficiency with which graphs can be embedded in the plane according to VLSI design rules. In this survey paper, I review some of the major accomplishments of VLSI layout theory and discuss how layout theory engendered the notion of area and volume-universal networks, such as fat-trees. These scalable networks offer a flexible alternative to the more common hypercube-based networks for interconnecting the processors of large parallel supercomputers. (This paper was an invited presentation at the 1989 Caltech Decennial VLSI Conference.)

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL Judy Little, Publications Coordinator	22b TELEPHONE (Include Area Code) (617) 253-5894 22c. OFFICE SYMBOL

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted
All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE

U.S. Government Printing Office: 1985—607-047

Unclassified

18. VLSI theory.

VLSI Theory and Parallel Supercomputing

Charles E. Leiserson

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

May 25, 1989

Abstract

Since its inception, VLSI theory has expanded in many fruitful and interesting directions. One major branch is layout theory which studies the efficiency with which graphs can be embedded in the plane according to VLSI design rules. In this survey paper, I review some of the major accomplishments of VLSI layout theory and discuss how layout theory engendered the notion of area and volume-universal networks, such as fat-trees. These scalable networks offer a flexible alternative to the more common hypercube-based networks for interconnecting the processors of large parallel supercomputers. (This paper was an invited presentation at the 1989 Caltech Decennial VLSI Conference.)

Keywords: Fat-trees, hypercubes, integrated circuits, interconnection networks, layout theory, parallel computing, supercomputing, universality, Thompson's model, tree-of-meshes, VLSI theory.

Ten years ago, Clark Thompson introduced a simple, graph-theoretic model for VLSI circuitry [22]. In Thompson's model, a circuit is a graph whose vertices correspond to active circuit elements and whose edges correspond to wires. A VLSI layout is a mapping of the graph to a two-dimensional grid, such that each vertex is mapped to a square region of the grid and each edge is mapped to a path in the grid. Unlike the classical notions of a graph embedding from mathematics, Thompson's model allows edges of a graph to cross over one another, like wires on an integrated circuit.

The interesting cost measure in VLSI is *area*. In Thompson's model, area can be measured as the number of grid points occupied by edges or vertices of the graph. Quickly, the minimum-area layouts for familiar graphs were catalogued. As shown in Figure 1, a mesh (two-dimensional array) with n vertices ($\sqrt{n}$ by $\sqrt{n}$) has $\Theta(n)$ area.[1] The normal way of drawing a complete binary tree (Figure 2a) has $\Theta(n \lg n)$ area, but the "H-tree" layout (Figure 2b) is much better: it has $\Theta(n)$ area. A hypercube, which is a popular interconnection network for parallel computers, requires considerably more area—$\Theta(n^2)$.

What causes a hypercube to occupy so much area? Although the size of a vertex grows slowly with the number of vertices in a hypercube, most of the area of a hypercube layout is devoted to *wires*. Figure 3 shows how the the problem of wiring a hypercube grows with the size of the hypercube. Wires are expensive, and wire area represents the capital cost of communication on a VLSI chip. By measuring communication costs in terms of the geometric concept of area, Thompson's model enabled a mathematical theory of communication in VLSI systems to develop.

From its origin, VLSI theory has expanded in many fruitful and interesting directions. Rather than attempting to describe the breadth of research in VLSI theory, however, I would like to revisit the accomplishments along one narrow path—layout theory—which I believe will have a fundamental impact on the architecture of large parallel supercomputers.

In his early work, Thompson discovered an important lower bound. The area of an n-vertex graph is related to its *bisection width*: the minimum number of edges that must be removed to partition the graph into two subgraphs

[1] The notation $\Theta(f(n))$ means a function that grows at the same rate as $f(n)$ to within a constant factor as n becomes large. The notation $O(f(n))$ means a function that grows no more quickly, and $\Omega(f(n))$ means a function that grows no more slowly. Formal definitions for these terms can be found in any textbook on analysis of computer algorithms.

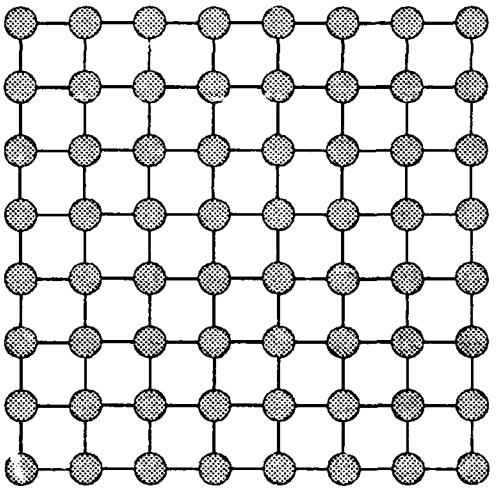
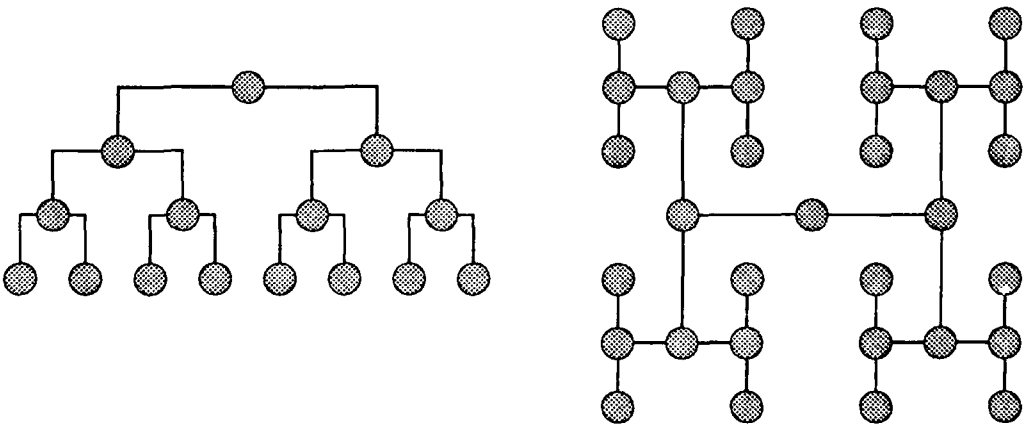


Figure 1: A mesh (two-dimensional array) on n vertices has a VLSI layout with $\Theta(n)$ area.



(a) (b)

Figure 2: A complete binary tree on n vertices laid out in the standard way (a) takes $\Theta(n \lg n)$ area, but an H-tree layout (b) requires only $\Theta(n)$ area.

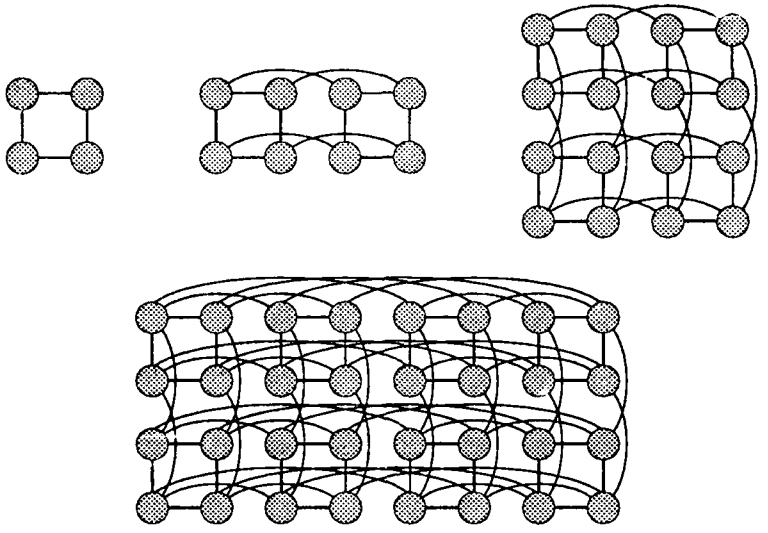


Figure 3: Illustrations (not layouts) of hypercubes on 4, 8, 16, and 32 vertices. Any layout of an n-vertex hypercube requries $\Omega(n^2)$ area.

of $n/2$ vertices (to within 1, if the number of vertices is odd). For example, an n-vertex mesh has a bisection width of $\sqrt{n}$. A complete binary tree has a bisection width of 1. A hypercube has a bisection width of $n/2$. Thompson proved that any layout of a graph with bisection width w requires $\Omega(w^2)$ area.

It turns out that a small bisection width does not lead immediately to a small-area layout. After all, if we take two $n/2$-vertex subgraphs, each with $\Theta(n^2)$ area, and connect them by a single edge, the resulting graph has a bisection width of 1 but still requires $\Theta(n^2)$ area. Leslie Valiant and I were able to show in independent work [24, 14, 15], however, that if there is a good *recursive decomposition* of a graph—one where we can keep subdividing the subgraphs without cutting many edges—then the graph has a small layout. For example, not only complete binary trees, but *any* binary tree, no matter how badly balanced, can be laid out in $O(n)$ area by a divide-and-conquer method. Valiant and I were also able to show that this method lays out any n-vertex planar graph in $O(n \lg^2 n)$ area. Later, Leighton was able to show that a variant of our method was optimal on any graph to within a $O(\lg^2 n)$

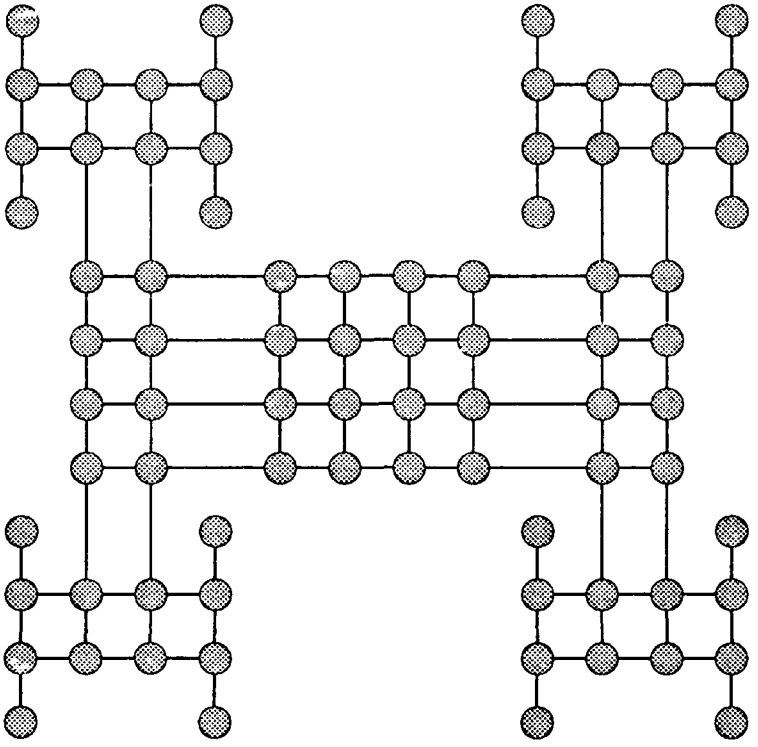


Figure 4: The tree-of-meshes graph.

factor in area [9].

Leighton also introduced an interesting graph which he called the *tree-of-meshes* graph, shown in Figure 4. He was able to prove that this graph requires $\Omega(n \lg n)$ area, thereby refuting a conjecture of mine that all planar graphs could be laid out in $O(n)$ area. It remains an open question in VLSI theory as to whether there exists a planar graph that requires $\Omega(n \lg^2 n)$ area, or if all planar graphs can be laid out in $O(n \lg n)$ area.

Numerous other results in layout theory have been obtained—too many to mention them all. Paterson, Ruzzo, and Snyder [18] and Bhatt and Leiserson [3] studied how to keep wires short while preserving small area. Valiant [24], Ruzzo and Snyder [21], and Dolev, Leighton, and Trickey [4] studied VLSI layouts in which wires are not allowed to cross. Three-dimensional integration

was studied by Rosenberg [20], Leighton and Rosenberg [13], and Greenberg and Leiserson [5]. Fault tolerance in wafer-scale circuits was studied by Rosenberg [19], Leighton and Leiserson [11, 10], and Greene and El Gamal [7]. The packaging of graphs into chips was studied by Leiserson [15] and Bhatt and Leiserson [2].

In fact, packaging constraints are analogous to the constraints in Thompson's model. At any level of packaging—chips, boards, backplanes, racks, or cabinets—manufacturing technology constrains the number of external connections from a package to be much smaller than the number of components within the package. In Thompson's model, a square region with side s can support $4s$ external connections, but it can contain s^2 vertices, which is considerably larger than $4s$ as s becomes large.

As an example of a result [15] in packaging, Figure 5 shows a novel way to package a complete binary tree using 4-pin packages of a single type. Each chip contains one internal node of the tree, with three external connections, and the remainder of the chip is packed as full as possible with a complete binary tree, with one external connection. To assemble a tree with twice as many leaves, we use two chips. We wire up one of the unconnected internal nodes on one of the chips as the parent of the two complete binary trees. We are left with a complete binary tree with twice as many leaves, plus one unconnected internal node. Thus, considering the two chips as a single unit, the structure is the same as the one with which we began. By repeating the process, we can recursively assemble a complete binary tree of arbitrarily large size.

The work in layout theory culminated with the development by Bhatt and Leighton [1] of a general framework for VLSI layout. They proposed a layout method with which they were able to obtain optimal or near-optimal layouts for many graph-embedding problems. Their method has three steps. First, recursively bisect the graph, forming a decomposition tree of the graph. Second, embed the graph in the tree-of-meshes graph (Figure 4), typically, with the vertices of the graph at the leaves of the tree-of-meshes graph. The meshes in the tree-of-meshes are used as crossbar switches for routing the edges of the graph. The layout of the graph is then obtained by looking at where the vertices and edges are mapped when the the tree-of-meshes graph is laid out according to known good layouts.

It seemed to me at the time that Bhatt and Leighton had solved nearly all the interesting open problems in VLSI layout theory. All new results in

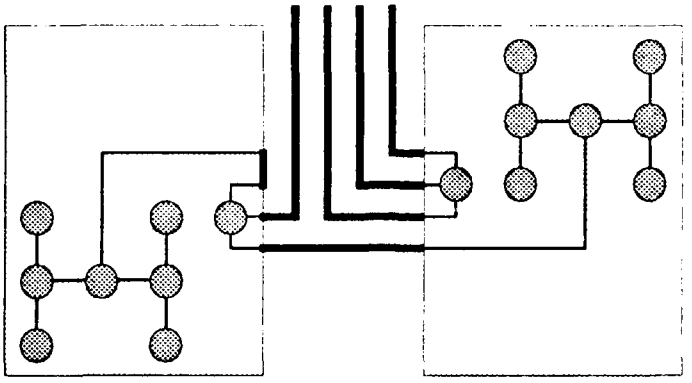


Figure 5: Packaging a complete binary tree.

the area would be little more than refinements of existing methods with no more real insights into the nature of interconnectivity. I turned my attention toward parallel computation, in which I had continued to be involved since my work with H. T. Kung on systolic arrays [8].

In fact, I was very much a proponent of special-purpose parallel computation over general-purpose parallel computation, largely as a result of my work on VLSI layout theory. After all, as Kung and I had shown, and as Kung has continued to forcefully demonstrate, many computations can be performed efficiently on simple linear-area structures such as one and two-dimensional arrays. These special-purpose networks have the nice property that they can be laid out so that processors are dense and packaging costs are minimized. Moreover, for many problems, they offer speedup which is linear in the number of processors in the systolic array.

General-purpose parallel computers, on the other hand, are typically based on interconnection networks, such as hypercubes, that are very costly for the computation they provide. For example, any hypercube network embedded in area A has at most $O(\sqrt{A})$ processors. The processors are therefore sparse in the embedding, and connections dominate the cost. Similar results can be shown for a three-dimensional VLSI model. Only $O(V^{2/3})$ processors of a hypercube network can fit in a volume V.

Hypercube networks do have a major advantage over many other networks for parallel computing, however. They are *universal*: a hypercube on n

processors can simulate any n-processor bounded-degree network in $O(\lg n)$ time. The simulation overhead is polylogarithmic (a polynomial of $\lg n$), an indication that the simulation is a parallel simulation. A polynomial overhead in simulation is less interesting, since $\Theta(n)$ overhead is easily obtained by a serial processor simulating each of the n processors in turn.

The proof that an n-processor hypercube is universal goes roughly as follows. Suppose we have a bounded-degree network R with n processors. Each processor can communicate with all its neighbors in unit time. The hypercube can simulate the network, therefore, by sending at most a constant number of messages from each processor, where each message contains the information that travels on one of the interconnections in R. It turns out, all messages can be routed on the hypercube to their destinations in $O(\lg n)$ time [23].

The notion of universality—the ability of one machine to efficiently simulate every machine in a class—is central to the origins of computer science. A universal machine is the computer theorist's idea of a general-purpose, as opposed to multipurpose, machine. A universal machine can do the function of any machine, just by programming it, or, in the case of parallel-processing networks, just by routing messages. A universal machine may not be the best machine for any given job, but it is never much worse than the best. The universality theorem for hypercubes does not say that a hypercube is the fastest network to build on n processors. What it says is that the fastest special-purpose network for any given problem can't be much faster.

From a VLSI theory standpoint, however, a special-purpose parallel machine has a clear advantage over a universal parallel machine. Packaging its network can cost much less. And although universality is a selling point, our economy favors machines that are cheap and efficient, even if they are not universal. (How many combination telephone-lawnmower-toothbrushes have been sold recently?) Special-purpose networks for parallel computation are much cheaper than hypercube networks. Thus, for a long time, I was skeptical about the cost-effectiveness of general-purpose parallel computing.

I changed my mind, however, and became an advocate general-purpose parallel computing when I started to look more closely at the traditional assumptions concerning universal networks. In fact, from a VLSI theory perspective, I discovered that hypercubes are not really "universal" at all! An n-processor hypercube may be able to efficiently simulate any n-processor bounded-degree network, but if we normalize by area instead of by number of

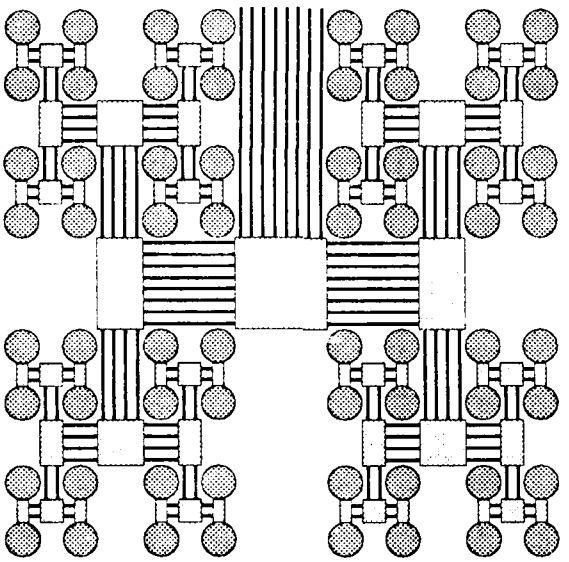


Figure 6: An area-universal fat-tree.

processors, we discover that an area-A hypercube cannot simulate all area-A networks efficiently. For example, since an area-A hypercube has only $\Theta(\sqrt{A})$ processors, it can't simulate an area-A mesh, which has $\Theta(A)$ processors, in polylogarithmic time. A network that is universal from a VLSI point of view should be a network that for a given area can efficiently simulate any other network of comparable area.

One such *area-universal* network is a fat-tree [16, 6], which is based on Leighton's tree-of-meshes graph. As shown in Figure 6, processors occupy the leaves of the tree, and the meshes are replaced with switches. Unlike a computer scientist's traditional notion of a tree, a fat-tree is more like a real tree in that it gets thicker further from the leaves. Local messages can be routed within subtrees, like phone calls in a telephone exchange, thereby requiring no bandwidth higher in the tree. The number of external connections from a subtree with m processors is proportional to $\sqrt{m}$, which is the perimeter of a region of area m. The area of the network is $O(n \lg^2 n)$, which is nearly linear in the number n of processors. Thus, the processors are packed densely in the layout.

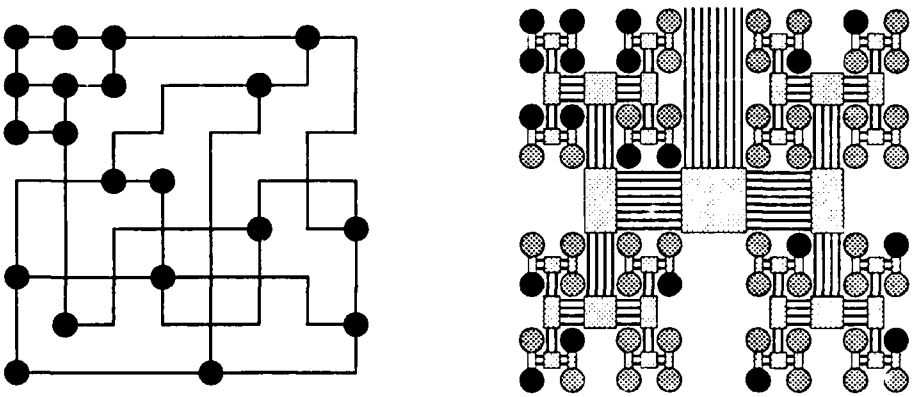


Figure 7: Any area-n network R can be efficiently simulated by an n-processor area-universal fat-tree.

Any network R that fits in a square of area n can be efficiently simulated by an area-universal fat-tree on n processors. To perform the simulation, we ignore the wires in R and map the processors of R to the processors of the fat-tree in the natural geometric way, as shown in Figure 7. As in the hypercube simulation, each wire of R is replaced by a message in the fat-tree. If we look at any m-processor subtree of the fat-tree, it simulates at most a region of area m in the layout of R. The number of wires that can leave this area-m region in R's layout is $O(\sqrt{m})$, and the fat-tree channel connecting to the root of the subtree has $\Theta(\sqrt{m})$ wires. Thus, the *load factor* of the channel, the ratio of the number of messages to channel bandwidth, is $O(1)$. It turns out that there are routing algorithms [16, 6, 12] that effectively guarantee that all messages are delivered in polylogarithmic time. (In fact, the algorithms can deliver messages near optimally even if the load factor is quite large.)

Similar universality theorems can be proved for three-dimensional VLSI models using *volume-universal* fat-trees. For a fat-tree to be universal for volume, however, the channel capacities must be selected differently from those in an area-universal network. Whereas the average growth rate of channels in an area-universal fat-tree is $\sqrt{2}$, the average growth rate in a volume-universal fat-tree is $\sqrt[3]{4}$.

In practice, of course, no mathematical rule governs interconnect technology. Most networks that have been proposed for parallel processing, such as meshes and hypercubes, are inflexible when it comes to adapting their topologies to the arbitrary bandwidths provided by packaging technology. The growth in channel bandwidth of a fat-tree, however, is not constrained to follow a prescribed mathematical formula. The channels of a fat-tree can be adapted to effectively utilize whatever bandwidths the technology can provide and which make engineering sense in terms of cost and performance. Figure 8 shows one variant of a fat-tree composed of two kinds of small switches: a three-connection switch and a four-connection switch. By choosing one of these two kinds of switches at each level of the fat-tree, the bandwidths of channels can be adjusted. If the three-connection switch is always selected, an ordinary complete binary tree results. If the four-connection switch is always selected, a butterfly network which is a relative of a hypercube, results. By suitably mixing these two kinds of switches, a fat-tree that falls between these two extremes can be constructed that closely matches the the bandwidths provided by the interconnect technology.

The notion of locality exploited by fat-trees is but one of three such notions that arise in the engineering of a parallel computer. The most basic notion of locality is exemplified by wire delay and measured in distance. Communication is speed-of-light limited. If this notion of locality dominates, the nearest-neighbor communication provided by a three-dimensional mesh is the best one can hope. For many systems, however, wire delay is dominated by the time it takes for logic circuits to compute their functions. The second notion of locality is exemplified by levels of logic circuits and measured in gate delays. Communication time is essentially limited by the number of switches a message passes through. From this point of view, structures with small diameters, such as hypercubes, seem ideal. In a routing network, however, a heavy load of messages can cause congestion, and the time it takes to resolve this congestion can dominate both wire and gate delays. Congestion is especially likely to occur in networks that make efficient use of packaging technology. The last notion of locality is exemplified by the congestion of messages leaving a subsystem and measured by load factor. From this standpoint, fat-trees offer provably good performance by a general-purpose network that can be packaged efficiently. Recent work [17] has shown that efficient parallel algorithms can be designed for this kind of network, as well.

Whatever the point of view, however, all three notions of locality must

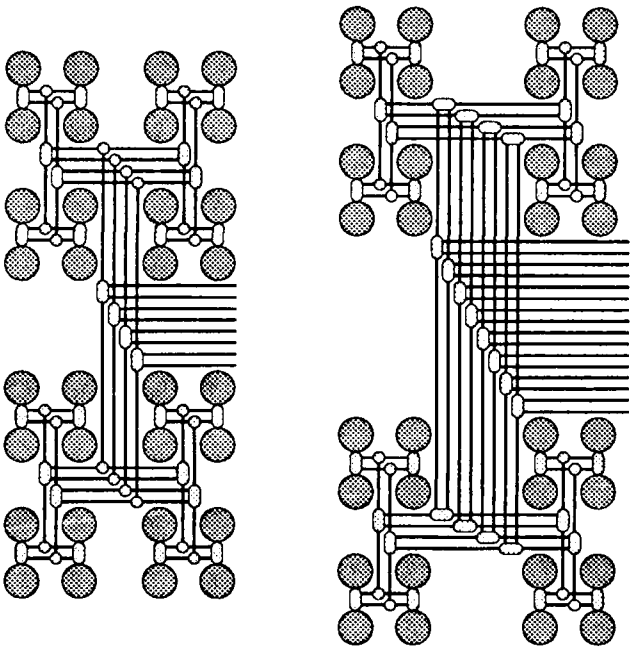


Figure 8: A scalable fat-tree.

guide the engineering and programming of very large machines. There are problems in the sciences that cry out for massive amounts of computation, most of which exhibit locality naturally: problems in astronomy, such as galaxy simulation; problems in biology, such as the combinatorics of DNA sequencing; problems in economics, such as market prediction; problems in aerospace, such as fluid-flow simulation; problems in earth, atmospheric, and ocean sciences, such as earthquake and weather prediction. To address these problems effectively, very large parallel computers must be constructed. Some of these computers may even be "building sized." To construct and program such large machines, however, locality must be exploited, and computer engineers must come to grips with the lessons of VLSI theory.

Acknowledgments

My research is supported in part by the Defense Advanced Research Projects Agency under Contract N00014-87-K-825 and in part by an NSF Presidential Young Investigator Award with matching funds provided by IBM Corporation, AT&T Bell Laboratories, and Xerox Corporation. I am grateful to these institutions for their continuing support.

References

[1] S. N. Bhatt and F. T. Leighton. A framework for solving VLSI graph layout problems. *Journal of Computer and System Sciences*, 28(2):300–343, 1984.

[2] S. N. Bhatt and C. E. Leiserson. How to assemble tree machines. *Advances in Computing Research*, 2:95–114, 1984.

[3] S. N. Bhatt and C. E. Leiserson. Minimizing the longest edge in a VLSI layout. 1981. Unpublished memorandum, MIT Laboratory for Computer Science.

[4] D. Dolev, F. T. Leighton, and H. Trickey. Planar embedding of planar graphs. *Advances in Computing Research*, 2:147–161, 1984.

[5] R. I. Greenberg and C. E. Leiserson. A compact layout for the three-dimensional tree of meshes. *Applied Mathematics Letters*, 1(2):171–176, 1988.

[6] R. I. Greenberg and C. E. Leiserson. Randomized routing on fat-trees. In *26th Annual IEEE Symposium on Foundations of Computer Science*, pages 241–249, 1985.

[7] J. W. Greene and A. El Gamal. Configuration of VLSI arrays in the presence of defects. *Journal of the ACM*, 31(4):694–717, 1984.

[8] H. T. Kung and C. E. Leiserson. Systolic arrays (for VLSI). In I. S. Duff and G. W. Stewart, editors, *Sparse Matrix Proceedings 1978*, pages 256–282, SIAM, 1979.

[9] F. T. Leighton. A layout strategy for VLSI which is provably good. In *14th Annual ACM Symposium on Theory of Computing*, pages 85–98, 1982.

[10] F. T. Leighton and C. E. Leiserson. A survey of algorithms for integrating wafer-scale systolic arrays. In *IFIP Conference on Wafer-Scale Integration*, pages 177–195, 1986.

[11] F. T. Leighton and C. E. Leiserson. Wafer-scale integration of systolic arrays. *IEEE Transactions on Computers*, C-34(5):448–461, 1985.

[12] F. T. Leighton, B. Maggs, and S. Rao. Universal packet routing algorithms. In *29th Annual IEEE Symposium on Foundations of Computer Science*, pages 256–271, 1988.

[13] F. T. Leighton and A. L. Rosenberg. Three-dimensional circuit layouts. *SIAM Journal on Computing*, 15(3):793–813, 1986.

[14] C. E. Leiserson. Area-efficient graph layouts for VLSI. In *21st Annual IEEE Symposium on Foundations of Computer Science*, pages 199–214, 1980.

[15] C. E. Leiserson. *Area-Efficient VLSI Computation. ACM Doctoral Dissertation Award Series*, MIT Press, Cambridge, Massachusetts, 1983.

[16] C. E. Leiserson. Fat-trees: universal networks for hardware-efficient supercomputing. *IEEE Transactions on Computers*, C-34(10):892–901, 1985.

[17] C. E. Leiserson and B. M. Maggs. Communication-efficient parallel algorithms for distributed random-access machines. *Algorithmica*, 3:53–77, 1988.

[18] M. S. Paterson, W. L. Ruzzo, and L. Snyder. Bounds on minimax edge length for complete binary trees. In *13th Annual ACM Symposium on Theory of Computing*, pages 293–299, 1981.

[19] A. L. Rosenberg. *The Diogenes approach to testable fault-tolerant networks of processors.* technical memorandum CS-1982-6.1, Department of Computer Science, Duke University, 1982.

[20] A. L. Rosenberg. Three-dimensional integrated circuitry. In H. T. Kung, R. Sproull, and G. L. Steele Jr., editors, *VLSI Systems and Computations*, pages 69–79, Computer Science Press, 1981.

[21] W. L. Ruzzo and L. Snyder. Minimum edge length planar embeddings of trees. In H. T. Kung, R. Sproull, and G. L. Steele Jr., editors, *VLSI Systems and Computations*, pages 119–123, Computer Science Press, 1981.

[22] C. D. Thompson. Area-time complexity for VLSI. In *Caltech Conference on Very Large Scale Integreation*, pages 495–508, 1979.

[23] L. G. Valiant. A scheme for fast parallel computation. *SIAM Journal on Computing*, 11(2):350–361, 1982.

[24] L. G. Valiant. Universality considerations in VLSI circuits. *IEEE Transactions on Computers*, C-30(2):135–140, 1981.

OFFICIAL DISTRIBUTION LIST

Director Information Processing Techniques Office Defense Advanced Research Projects Agency 1400 Wilson Boulevard Arlington, VA 22209	2 copies
Office of Naval Research 800 North Quincy Street Arlington, VA 22217 Attn: Dr. R. Grafton, Code 433	2 copies
Director, Code 2627 Naval Research Laboratory Washington, DC 20375	6 copies
Defense Technical Information Center Cameron Station Alexandria, VA 22314	12 copies
National Science Foundation Office of Computing Activities 1800 G. Street, N.W. Washington, DC 20550 Attn: Program Director	2 copies
Dr. E.B. Royce, Code 38 Head, Research Department Naval Weapons Center China Lake, CA 93555	1 copy