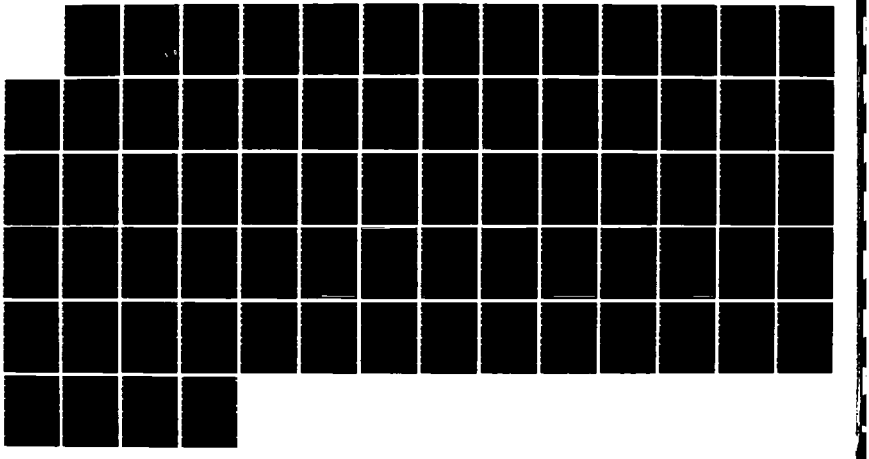
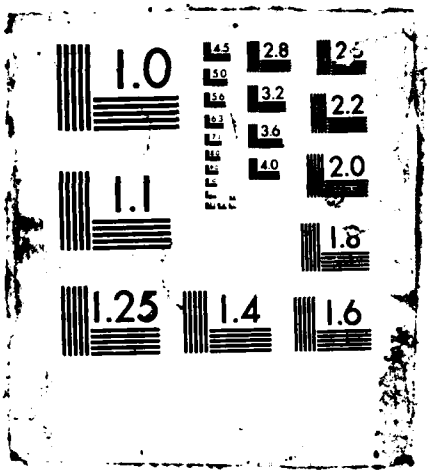
AD-A179 037 A COHERENT VLSI DESIGN ENVIRONMENT(U) MASSACHUSETTS INST OF TECH CAMBRIDG MICROSYSTEMS RESEARCH CENTER P PENFIELD ET AL 30 SEP 86 1/1

UNCLASSIFIED F/G 9/5 NL




1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.8
3.2
3.6
4.0

A COHERENT VLSI DESIGN ENVIRONMENT

Semiannual Technical Report for the period April 1, 1986 to September 30, 1986

Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

Principal Investigators: Paul Penfield, Jr. (617) 253-2506
William J. Dally (617) 253-6043
Lance A. Glasser (617) 253-4677
Thomas F. Knight, Jr. (617) 253-7807
Charles E. Leiserson (617) 253-5833
John L. Wyatt, Jr. (617) 253-6718
Richard E. Zippel (617) 253-6028

This research was sponsored by Defense Advanced Research Projects Agency (DoD), through the Office of Naval Research under Contract N00014-80-C-0622.

TABLE OF CONTENTS

Selected Publications (starting after page 12)

B. M. Maggs, "Communication-Efficient Parallel Graph Algorithms," M.S. thesis, Department of Electrical Engineering and Computer Science, MIT, Cambridge, MA, May 1986. *

T. S. Hohol, "RELIC: A Reliability Simulator for Integrated Circuits," M.S. thesis, Department of Electrical Engineering and Computer Science, MIT, Cambridge, MA, May 1986. Also MIT VLSI Memo No. 86-324, May 1986. *

F. M. Maley, "Compaction with Automatic Jog Introduction," M.S. thesis, Department of Electrical Engineering and Computer Science, MIT, May 1986. Also, MIT VLSI Memo No. 86-316, May 1986. *

T. Leighton and P. Shor, "Tight Bounds for Minimax Grid Matching, with Applications to the Average Case Analysis of Algorithms," Proceedings of the 18th ACM Symposium on Theory of Computing, Berkeley, CA, May 1986, pp. 91-103. Also MIT VLSI Memo No. 86-320, June 1986. *

F. Thompson Leighton, "A Survey of Problems and Results for Channel Routing," Aegean Workshop on Computing, Greece, July 1986.

T. H. Cormen and C. E. Leiserson, "A Hyperconcentrator Switch for Routing Bit-Serial Messages," Proceedings, 1986 IEEE International Conference on Parallel Processing, St. Charles, IL, August 19-22, 1986, pp. 721-728. *

C. E. Leiserson and B. M. Maggs, "Communication-Efficient Parallel Graph Algorithms," Proceedings, 1986 IEEE International Conference on Parallel Processing, St. Charles, IL, August 19-22, 1986, pp. 861-868. *

A. T. Sherman, "Cryptology and VLSI (a two-part dissertation): I. Detecting and Exploiting Algebraic Weaknesses in Cryptosystems; II. Algorithms for Placing Modules on a Custom VLSI Chip," Ph.D. thesis, Department of Electrical Engineering and Computer Science, MIT, October 1986. Also, to appear as MIT VLSI Memo No. 86-343, October 1986. *

William J. Dally, "A High Performance VLSI Quaternary Serial Multiplier," to appear as MIT VLSI Memo No. 86-344, October 1986. *

William J. Dally, "Wire-Efficient VLSI Multiprocessor Communication Networks," to appear as MIT VLSI Memo No. 86-345, October 1986. *

William J. Dally, "Lazy Event-Driven Simulation," to appear as MIT VLSI Memo No. 86-346, October 1986.

Sandeep Bhatt, Fan Chung, Tom Leighton, and Arnold Rosenberg, "Optimal Simulations of Tree Machines," Proceedings, 27th IEEE Conference on Foundations of Computer Science, Portland, OR, October 1986, pp. 274-282.

Richard E. Zippel, Paul Penfield, Jr., Lance A. Glasser, Charles E. Leiserson, John L. Wyatt, Jr., F. Thomson Leighton, and Jonathan Allen, "Recent Results in VLSI CAD at MIT," to appear in Proceedings Fall Joint Computer Conference, Dallas, TX, November 1986. Also MIT VLSI Memo No. 86-341, October 1986.

A. V. Goldberg and S. A. Plotkin, "Parallel (Δ+1) Coloring of Constant-Degree Graphs," unpublished manuscript.

---

* Abstract only. Complete version available from Microsystems Research Center, Room 39-321, MIT, Cambridge, MA 02139; telephone (617) 253-8138.

RESEARCH OVERVIEW

The report covers the period from April 1, 1986 through September 30, 1986. THe research discussed here is described in more detail in several published and unpublished reports cited below.

The CAD frame Schema has been outfitted with a uniform protocol or aggregate data structures, which includes new control structures for examining and searching through the data structures. Circuit simulators installed in Schema can now use their own scratchpad, and call a variety of tools to deal with the pertinent data structures.

Waveform bounding results are now beginning to be available for ECL circuits, similar to those reported earlier for MOS interconnect and gates. The principal problem is that a vastly different gate model appears to be necessary for rising and falling transients. So far results within a few percent of SPICE have been obtained for some cases.

A new procedure for calculating maximum frequency response of networks made from classes of devices with certain types of small-signal models, connected in arbitrary ways that conserve real and reactive power, has been worked out. This activity required the development of a new procedure to test whether 0 is in the numerical range of a non-Hermitian matrix.

The first version of RELIC, a reliability simulator, is now running and has been reported on. Simultaneous simulation of hot carriers, metal migration, and time dependent breakdown effects have been performed. Work is under way planning for the next version.

Improved algorithms have been found for many problems of importance in VLSI design and architecture. These include advances in channel routing, network simulation with hypercubes, fault-tolerant construction of networks, communication-efficient parallel graph algorithms, circuit verification, message-routing, network flow, graph coloring, and intelligent backtracking. Universality results for volume-universal networks have been strengthened. A thesis published during this period contains the most complete description of the PI placement and interconnect program. Automatic jog insertion during one-dimensional compaction was the topic of another thesis.

A lazy event-driven simulator, that delays expensive calculations until the results are needed (if ever) has been demonstrated to achieve substantial improvements in performance. Design has begun on a new routing chip to support bidirectional communication at a single port. The logic design is tighter and the expected speed is greater than prior routing chips.

THE DESIGN OF SCHEMA

As mentioned in the previous report, most of our efforts on Schema over the past few months have been spent fixing bugs and generally improving the details of the system. These improvements fall into three major areas: new aggregate data structures for dealing with large numbers of objects, new simulation data structures to improve efficiency, and improvements to the schematic entry system so that it may be used for TTL designs.

An aggregate data structure is a data structure that can contain some number of elements. Examples of such structures are lists, heaps, binary trees, quad trees and so on. In Schema, each of these data structures is implemented as a flavor. A uniform protocol has been developed that these data structures must obey. This protocol includes new control structures for examining and searching through the data structures. This approach makes it unnecessary for code developers to re-implement these structures, and allows them to change from one to another easily as their needs change. In particular, this has allowed us to begin using 2-D trees in the schematic editor to dramatically speed up the wiring interface. It also dramatically simplifies the layout editing tools. This work was done by Feng Zhao.

In a similar vein we have recast the way simulations and circuit analysis are done to provide each such tool its own scratchpad for notes about the circuit. The tools provided automatically flatten the hierarchical topology, create the scratchpad arrays, and define accessing macros and new control structures for searching the arrays. Again this has made the writing of these types of programs much easier and less error prone. Using these tools Lance Glasser, John Wroclawski, and Rich Zippel developed a small package for computing the admittance matrix of a multiport circuit.

Our colleagues at Harris Corporation (George Clark, Jim Leninger, Mike MacDonald and Adam Zilinskas) have been extending the schematic entry tools to allow Schema's use in designing TTL circuits. They have introduced busses into the schematic editor, cleaned up the wiring model, developed an association between gates and packages, and developed a graphical icon editor for user-defined modules. The editor is now quite comfortable in doing both bottom-up and top-down schematic designs.

THE WAVEFORM BOUNDING APPROACH TO TIMING ANALYSIS

Our work over the last six months has been concentrated on reworking the waveform bounding results of Penfield et. al., originally created for timing analysis of MOS chips, into a form useful for delay estimation in ECL circuits. A smaller effort has also been devoted to the circuit design and layout of a parallel analog MOS VLSI chip for early vision applications.

Peter O'Brien has continued to work on ECL delay estimation for his S.M. thesis, in cooperation with the Digital Equipment Corporation. The final product we envision is a timing analyzer, written in C, for use on ECL standard cell designs. The goal remains accurate estimation of signal propagation delay without resorting to computationally intensive numerical solution of the network equations. In apparent contrast to the earlier MOS-related work, the largest technical problem we've encountered is not estimating signal delay in interconnect, but rather accurately modelling the electrical behavior of the gates themselves. The ECL gates we've studied act like voltage sources with internal resistance when pulling a line high, but like almost ideal current sources when pulling it low. The latter case does not fit easily into the framework of the Penfield approach, which is tailored to voltage-source drives.

Our modelling progress to date consists of a reasonably simple macromodel for the gates themselves, and an approximation to the driving-point impedance of loaded, branched metal interconnect lines that lets us translate a current-source drive into an approximate resulting voltage waveform at the output of the driving gate. This combined procedure gives delay results that agree with detailed SPICE simulations to around 4%, with a savings of about 3 orders of magnitude in computer time. The only fundamental hurdle remaining is to find a simple but accurate way of approximating the voltage waveform at the input of any driven gate by a simplified waveform, such as a saturated ramp, that permits closed-form calculation of the gate macromodel's response.

On our other project, design of the analog depth-interpolation chip, has progressed to the point that we expect to send the first test chip to MOSIS in November. This design contains a 4x4 pixel array that is directly accessible from the pins, and test structures with each of the subcircuits needed for the final system, e.g., A/D and D/A converters and differential amplifiers. We'll tell you how it worked in the next progress report.

HIGH PERFORMANCE CIRCUIT DESIGN

Maximum Frequency of Oscillation: We have investigated limitations on the frequency response of networks constructed out of components specified by their small signal models. Tellegen's theorem and the conservation of complex power is used to find the maximum frequency of oscillation. Transistor and negative resistance amplifier examples have been developed. This work has application to the development and comparison of competing device technologies. As part of this research effort we have developed a numerical test to determine whether zero is an element of the numerical range of a non-hermitian matrix Y. In other words, we have developed a test to find when the inner product $\langle v, Yv \rangle$ includes the origin for $\langle v, v \rangle = 1$.

Modeling of magnetostatic couplers for CMOS VLSI chips: Analytical models for on-chip magnetostatic coupling antennas have been developed and validated experimentally. These models take into account substrate resistivity, and the distributed effects of metal resistance, capacitance, and inductance.

RELIC I: The first experimental version of RELIC--our circuit reliability simulator--has been completed and applied to test circuits. Simultaneous simulation of hot carriers, metal migration, and time dependent breakdown effects have been performed. Planning for a second generation version is now underway.

ARCHITECTURAL DESIGN

Professor Leighton has been working on problems in channel routing, network simulation and fault-tolerant construction of networks. In the area of channel routing, he and his student Bonnie Berger have found improved algorithms for routing in multilayer channels. The new algorithms come very close to achieving optimal routings for many problems and achieve the best worst-case performance of any known algorithm. They also improved the best known lower bounds on what one can hope to achieve in the new unit-vertical-overlap model of routing. A description of this work can be found in the unpublished paper, "Nearly Optimal Algorithms and Bounds for Multilayer Channel Routing," by Bonnie Berger, Martin Brady, Donna Brown, and Tom Leighton.

In the area of network simulations, Professor Leighton (with others) has discovered a way in which the hypercube can be used to simulate any binary tree of the same size with constant overhead. Previously, such constant-overhead simulation results were known only for the complete binary tree and grid networks. Simulation of arbitrary (i.e., not perfectly balanced) binary trees seemed to require $O(\log N)$ overhead where N is the number of nodes in the hypercube. The results are significant because binary trees arise in a variety of applications (e.g., numerical calculations and formula evaluation) and hypercubes appear to be the network of choice for many parallel architectures (e.g., the connection machine and the cosmic cube). The result is also important from a design point of view since a designer can effectively build a tree machine for every possible binary tree by simply building a single hypercube. The details of this work are described in a paper cited below by S. Bhatt, F. Chung, T. Leighton, and A. Rosenberg.

In the area of fault-tolerant construction of networks, Professor Leighton has been studying the problem of configuring a functioning network from a larger network of the same kind that contains random faults. This problem arises in at least two important contexts. First, in the area of wafer-scale integration of systolic arrays, the designer is presented with a wafer of cells (some of which are faulty) arranged in a grid pattern. The object is to link the functioning cells together into an array using wires that are as short as possible. Professor Leighton recently discovered a new algorithm for solving this problem which has provably superior performance (on average) to previous algorithms. The details are described in papers cited below by T. Leighton and P. Shor, and by T. Leighton and C. E. Leiserson.

The same problem also arises in the design of very large hypercube networks. Such networks are likely to experience faults over time. After each fault (or set of faults) the network must reconfigure itself to avoid the fault, making maximum use of the functioning processors but without introducing delays because of long wires or congestion in the functioning communication channels. Very recent work by Professor Leighton and his student Mark Newman includes a scheme for constructing a functioning $N/2$ node hypercube from an N node hypercube that contains up to $(1/2 - \varepsilon)N$ faulty cells. The algorithm works locally and quickly, and preserves the locality of node to node connections with high probability. The details of this preliminary research will be presented at the MIT VLSI Research Review, December 15, 1986.

Bruce Maggs completed his Master's thesis on communication-efficient parallel graph algorithms. The thesis shows that many graph problems can be solved in parallel, not only in a polylogarithmic number of steps, but with efficient communication at each step. The communication requirements of an algorithm are measured in a new model of a parallel computer called the distributed random access machine (DRAM), which is an abstraction of volume-universal networks such as fat-trees. The thesis includes joint work with Professor Leiserson.

Cindy Phillips and Charles Leiserson have developed an efficient parallel algorithm for determining the connected components of bounded degree planar graphs. The algorithm requires $O(\lg n)$ randomized time or $O(\lg n \lg^* n)$ deterministic time on an n-processor exclusive-read exclusive-write (EREW) PRAM. A similar algorithm runs in $O(\lg n \lg \lg n)$ deterministic time for any graph with $O(n)$ genus. Previous algorithms for this problem in the EREW model require $\Theta(\lg^2 n)$ time in the worst case. The key to efficiency of the algorithm is the ability to quickly grow a spanning tree for such graphs. Thus the techniques of the algorithm may improve the running times of other graph problems of interest in vision and VLSI applications.

Alexander Ishii and Professor Leiserson have developed an $O((|F| + |T|)K)$ time algorithm for verifying the correct operation of a computation on a level-clocked synchronous circuit, where F and T are respectively the functional blocks and data transmission paths of the circuit and K is the length of the computation. The algorithm is unique in its ability to qualify the effect of timing errors on the final output of a circuit. Only errors which potentially change the circuit output will result in circuit disqualification. Circuits are represented as graphs and are analyzed using a formal semantic model of circuit functionality which is sufficiently powerful to prove the correctness of the algorithm. Necessary and sufficient conditions for circuit correctness are given, along with precise specifications of the algorithms abilities and limitations.

James K. Park has been working on a simple deterministic nonline message-routing algorithm for fat trees. He has also discovered modular layouts for the area- and volume-universal fat trees with constant size switches.

Jeffrey Mark Siskind has been investigating intelligent backtracking algorithms for pure Prolog. The basic strategy consists of compiling first order horn-clause logic programs into an infinite SAT problem schema which is then given to a SAT solver which uses dependency directed backtracking techniques to prune the search space. In addition to handling pure Prolog, the search techniques can be extended to handle the solution of systems of simple linear inequalities which are combined with logical connectives. This extension has application in the area of VLSI layout compaction. Many graph algorithms come into play in the overall strategy, including union-find, connected-components, and shortest-path. As efficient parallel algorithms are known for many of these graph problems these techniques show promise of being amenable to efficient implementation on parallel computers.

Ronald Greenberg and Professor Leiserson have been considering ways of strengthening universality results for volume-universal (and area-universal)

networks. Previous results have shown that given any network occupying a particular volume, it is possible to select a network of essentially the same volume from the class referred to as fat-trees, which can efficiently simulate the given network. Current efforts focus on establishing, for each particular volume, a single network which can efficiently simulate all other networks of the same volume. Such a network could be viewed as a best possible general-purpose arrangement of processors and interconnect given a particular volume constraint.

In June, 1986, Tom Cormen submitted his master's thesis, entitled "Concentrator Switches for Routing Messages in Parallel Computers." The thesis contains two concentrator switch designs of interest. The first is for a combinational hyperconcentrator chip that has a highly regular layout in either ratioed nMOS or domino CMOS. The circuit takes advantage of the relatively fast performance of large fan-in NOR gates in MOS technologies. A signal incurs exactly $2 \lg n$ gate delays in passing through an n-by-n hyperconcentrator chip. The area of this switch is $\Theta(n^2)$. The second switch of interest is an (N, M, α) partial concentrator switch based on a recent technique for sorting on a mesh. This switch is also combinational, uses the hyperconcentrator chip as a subcircuit, requires at most $2\sqrt{N} + \lceil (\lg N)/2 \rceil$ data pins per chip, has $\alpha = 1 - O(N^{3/4}/M)$, and can be packaged in three dimensions with volume $O(N^{3/2})$. A signal incurs at most $(7 \lg N)/2 + O(1)$ gate delays in passing through this switch.

Alan T. Sherman, a student of Professor Ronald L. Rivest, finished his Ph.D. dissertation entitled "Cryptology and VLSI (a two-part dissertation): I. Detecting and Exploiting Algebraic Weaknesses in Cryptosystems; II. Algorithms for Placing Modules on a Custom VLSI Chip." The thesis includes the most complete description published to date of the PI placement and interconnect system.

Shlomo Kipnis together with Joe Kilian and Professor Leiserson have been investigating the power of multipin nets. They established a correspondence between multipin permutation architectures and difference covers for sets of permutations. For example, to realize all the cyclic shifts permutations on n chips in one clock cycle, only $\sqrt{n}$ pins per chip are required. Using 2-pin nets, $n - 1$ pins per chip are required. They also show that $O(\sqrt{n})$ pins suffice to realize any abelian group of permutations and that any general group of permutations requires $O(\sqrt{(n \backslash \lg n)})$ pins per chip. Their results can be extended to realizing permutations in more than one clock cycle.

Miller Maley completed his Master's thesis on the topic of VLSI layout compaction with automatic jog insertion. It applies results of Maley and Leiserson on single-layer routing to obtain an efficient algorithm for one-dimensional compaction that treats wires as flexible objects, inserting all jog points that help to reduce the circuit size. Using the techniques of homotopy theory, Maley solidified and extended the theory of single-layer wire routing that underlies his algorithms for compaction and single-layer wire routing. As a result, these algorithms can now be generalized and be rigorously proven correct.

Andrew Goldberg has been working on algorithms, including parallel algorithms. He has continued his work on network flow problems and succeeded in generaliz-

ing the approach on maximum flow problem developed last year to obtain better sequential, parallel and distributed algorithms for the minimum-cost flow problem, a problem of broad importance including applications to VLSI. Together with Professor Leiserson, he investigated techniques for implementing parallel graph algorithms on highly parallel computers. Goldberg implemented the maximum flow algorithm on the Connection Machine while a summer intern at Thinking Machines Corporation and showed these techniques to be efficient.

Andrew Goldberg and Serge Plotkin investigated parallel algorithms that use a linear number of processors to solve certain graph coloring problems and the maximal independent set problems. They have developed efficient algorithms that work on a wide class of graphs. For example, they have shown that a maximal independent set in a constant-degree graph with n nodes can be found in $O(\log^* n)$ time.

SIMULATION AND COMMUNICATIONS

A lazy event-driven simulator has been developed, that reduces the computation required to simulate a circuit by avoiding the evaluation of modules whose outputs will later be ignored. A prototype has been implemented in ZetaLISP on a Symbolics 3640 workstation. Simulations of test circuits showed reductions in simulation time ranging from 30% to 50% with greater savings for large circuits. Even better performance is expected for larger circuits.

The design of a new VLSI routing chip, the bidirectional torus router (BTR), has begun. The BTR incorporates many of the features of the previous design by Professor Dally, the Torus Routing Chip (TRC), including wormhole routing and a deadlock-free routing algorithm based on virtual channels. Like the TRC, the BTR will be self-timed.

Unlike the TRC, the BTR will support bidirectional communication on a single set of wires, will support adaptive routing, will have a special end of message character to avoid bit-stuffing, and will provide parity checking. The BTR also incorporates a number of design changes that will increase its performance and reduce its area. The crossbar switch of the TRC is being eliminated in favor of a simpler dedicated switch in the BTR. This greatly reduces chip area while speeding up the path from input to output. The self-timed input controller, one of the slowest parts of the TRC, has also been redesigned. The BTR controller is much smaller than the TRC controller and has only 8 gate delays from input to output. These improvements should give the BTR five times the performance of the TRC.

PUBLICATIONS

P. Bassett, L. A. Glasser, and R. Rettberg, "Dynamic Delay Adjustment: A Technique for High-Speed Asynchronous Communication," Proc. Fourth MIT Conference on Advanced Research in VLSI, Cambridge, MA, April 7-9, 1986, pp. 219-232.

P. R. O'Brien and J. L. Wyatt, Jr., "Signal Delay in ECL Interconnect," Proceedings, 1986 International Symposium on Circuits and Systems, Santa Clara, CA, pp. 755-758, May 5-7, 1986. Also MIT VLSI Memo No. 86-296, February 1986.

T. S. Hohol, "RELIC: A Reliability Simulator for Integrated Circuits," M.S. thesis, Department of Electrical Engineering and Computer Science, MIT, Cambridge, MA, May 1986. Also MIT VLSI Memo No. 86-324, May 1986.

B. M. Maggs, "Communication-Efficient Parallel Graph Algorithms," M.S. thesis, Department of Electrical Engineering and Computer Science, MIT, Cambridge, MA, May 1986.

F. M. Maley, "Compaction with Automatic Jog Introduction," M.S. thesis, Department of Electrical Engineering and Computer Science, MIT, Cambridge, MA, May 1986. Also, MIT VLSI Memo No. 86-316, May 1986.

T. Leighton and P. Shor, "Tight Bounds for Minimax Grid Matching, with Applications to the Average Case Analysis of Algorithms," Proceedings of the 18th ACM Symposium on Theory of Computing, Berkeley, CA, May 1986, pp. 91-103. Also MIT VLSI Memo No. 86-320, June 1986.

A. V. Goldberg and R. E. Tarjan, "A New Approach to the Maximum Flow Problem," Proceedings of the 18th ACM Symposium on Theory of Computing, Berkeley, CA, May 1986, pp. 136-146.

A. L. Robinson, L. A. Glasser, and D. A. Antoniadis, "A Simple Interconnect Delay Model for Multilayer Integrated Circuits," VLSI Multilevel Interconnect Conference, June 9-11, 1986. Also MIT VLSI Memo No. 85-223, January 1985.

J. L. Wyatt, Jr., C. A. Zukowski, and Paul Penfield, Jr., "Step Response Bounds for Large-Scale Linear Systems Described by M-Matrices, with VLSI Application," SIAM Conference on Linear Algebra in Signals, Systems and Control, Boston, MA, August 12-14, 1986; abstract appeared on pp. A35-A36 of Conference Program.

T. H. Cormen and C. E. Leiserson, "A Hyperconcentrator Switch for Routing Bit-Serial Messages," Proceedings, 1986 IEEE International Conference on Parallel Processing, St. Charles, IL, August 19-22, 1986, pp. 721-728.

C. E. Leiserson and B. M. Maggs, "Communication-Efficient Parallel Graph Algorithms," Proceedings, 1986 IEEE International Conference on Parallel Processing, St. Charles, IL, August 19-22, 1986, pp. 861-868.

T. Leighton and A. Rosenberg, "Three-Dimensional Circuit Layouts," SIAM J. Computing, vol. 15, No. 3, August 1986, pp. 793-813. An early version appeared as VLSI Memo No. 84-182, June 1984.

A. T. Sherman, "Cryptology and VLSI (a two-part dissertation): I. Detecting and Exploiting Algebraic Weaknesses in Cryptosystems; II. Algorithms for Placing Modules on a Custom VLSI Chip," Ph.D. thesis, Department of Electrical Engineering and Computer Science, MIT, October 1986. Also, to appear as MIT VLSI Memo No. 86-343, October 1986.

William J. Dally, "A High Performance VLSI Quaternary Serial Multiplier," to appear as MIT VLSI Memo No. 86-344, October 1986.

William J. Dally, "Wire-Efficient VLSI Multiprocessor Communication Networks," to appear as MIT VLSI Memo No. 86-345, October 1986.

William J. Dally, "Lazy Event-Driven Simulation," to appear as MIT VLSI Memo No. 86-346, October 1986.

Sandeep Bhatt, Fan Chung, Tom Leighton, and Arnold Rosenberg, "Optimal Simulations of Tree Machines," Proceedings, 27th IEEE Conference on Foundations of Computer Science, Portland, OR, October 1986, pp. 274-282.

Richard E. Zippel, Paul Penfield, Jr., Lance A. Glasser, Charles E. Leiserson, John L. Wyatt, Jr., F. Thomson Leighton, and Jonathan Allen, "Recent Results in VLSI CAD at MIT," to appear in Proceedings, Fall Joint Computer Conference, Dallas, TX, November 1986. Also MIT VLSI Memo No. 86-341, October 1986.

T. Leighton and C. E. Leiserson, "A Survey of Algorithms for Integrating Wafer-Scale Systolic Arrays," Wafer Scale Integration, G. Saucier and J. Trilhe, editors, Elsevier Science Publishers B.V., IFIP, 1986, pp. 177-195. Portions of this work are based on the paper "Wafer-Scale Integration of Systolic Arrays" by Leighton and Leiserson, which appeared in IEEE Transactions on Computers, Vol. C-34, No. 5, pp. 448-461, May 1985. Portions of this paper will also appear in VLSI Systems Architecture, E. Swartzlander, Jr., ed., Marcel-Dekker, Inc., 1987. An early version of this paper appeared as VLSI Memo No. 82-101, April, 1982 and later a revised version appeared as VLSI Memo No. 85-272, October 1985.

F. Chung, T. Leighton and A. Rosenberg, "Embedding Graphs in Books: A Layout Problem With Applications to VLSI Design," to appear in SIAM J. Algebraic and Discrete Methods, vol. 8, No. 1, January 1987. Also MIT VLSI Memo No. 85-235, March 1985.

T. Bui, S. Chaudhuri, T. Leighton and M. Sipser, "Graph Bisection Algorithms with Good Average Case Behavior," to appear in Combinatorica. Also MIT VLSI Memo No. 85-236, March 1985.

A. V. Goldberg and S. A. Plotkin, "Parallel (Δ+1) Coloring of Constant-Degree Graphs," unpublished manuscript.

Bonnie Berger, Martin Brady, Donna Brown, and Tom Leighton, "Nearly Optimal Algorithms and Bounds for Multilayer Channel Routing," unpublished manuscript.

TALKS WITHOUT PROCEEDINGS

F. Thomson Leighton, "Reconfiguration Strategies for Wafer-Scale Integration," IFIP Workshop on Wafer-Scale Integration, France, March 1986.

J. L. Wyatt, Jr., "Step Response Bounds for Large-Scale Linear Systems Governed by M-Matrices, with Application to Timing Analysis of VLSI Circuits," Laboratory for Information and Decision Systems Seminar, MIT, Cambridge, MA, April 1986.

D. L. Standley and J. L. Wyatt, Jr., "Improved Signal Delay Bounds for RC Tree Networks," MIT Spring 1986 VLSI Research Review, Cambridge, MA, May 19, 1986.

F. Thompson Leighton and Peter Shor, "Improved Algorithms for Wafer-Scale Integration of 2-D Systolic Arrays," MIT Spring 1986 VLSI Research Review, Cambridge, MA, May 19, 1986.

T. Hohol and L. A. Glasser, "RELIC: A Reliability Simulator for Integrated Circuits," MIT Spring 1986 VLSI Research Review, Cambridge, MA, May 19, 1986.

C. E. Leiserson and B. M. Maggs, "Communication-Efficient Parallel Graph Algorithms," MIT Spring 1986 VLSI Research Review, Cambridge, MA, May 19, 1986.

Alan Sherman, "Algorithms for Placing Modules on a Custom VLSI Chip," MIT Spring 1986 VLSI Research Review, Cambridge, MA, May 19, 1986.

J. L. Wyatt, Jr., "Signal Propagation Delay in RC Models for Integrated Circuit Interconnect-Linear Theory and Nonlinear Extensions," Engineering Foundation Conference on Qualitative Methods for the Analysis of Nonlinear Dynamics, New England College, Henniker, NH, June 1986.

J. L. Wyatt, Jr., "The Practical Man's Introduction to Signal Delay in MOS Interconnect," Summer Course on MOS IC Design, MIT, Cambridge, MA, June 1986; also July 1986, Fairchild Semiconductor Corp., Palo Alto, CA.

F. Thompson Leighton, "A Survey of Problems and Results for Channel Routing," Aegean Workshop on Computing, Greece, July 1986.

Communication-Efficient Parallel Graph Algorithms

by

Bruce MacDowell Maggs

Submitted to the Department of
Electrical Engineering and Computer Science
on May 16, 1986
in Partial Fullfillment of the
Requirements of the Degree of

Master of Science
in
Electrical Engineering and Computer Science

Abstract—Communication bandwidth is a resource ignored by most parallel random-access machine (PRAM) models. This thesis shows that many graph problems can be solved in parallel, not only with polylogarithmic performance, but with efficient communication at each step of the computation. The communication requirements of an algorithm are measured in a restricted PRAM model called the *distributed random-access machine* (DRAM), which can be viewed as an abstraction of volume-universal networks such as fat-trees. In this model, communication cost is measured in terms of the congestion of memory accesses across cuts of the machine. We demonstrate that the "recursive doubling" technique frequently used in PRAM algorithms is wasteful of communication resources, and that "recursive pairing" can be used to perform many of the same functions more efficiently. We generalize the prefix computation on linear lists to trees and show that these *treefix* computations, which can be performed in a communication-efficient fashion using a variant of the tree contraction technique of Miller and Reif, simplify many parallel graph algorithms in the literature.

Thesis Supervisor: Professor Charles E. Leiserson
Title: Associate Professor

This thesis describes joint work with Professor Charles E. Leiserson. This research was supported in part by the Defense Advanced Research Projects Agency under Contract N00014-80-C-0622. Charles Leiserson is supported in part by an NSF Presidential Young Investigator Award. Bruce Maggs is supported in part by an NSF Fellowship.

VLSI Memo No. 86-324
June 1986

RELIC: A RELIABILITY SIMULATOR FOR INTEGRATED CIRCUITS

Teresa S. Hohol

Abstract

RELIC is a reliability simulator developed to analyze the stress and wear on MOS VLSI chips. The analysis, which is done in the design phase, is useful in designing for worst-case reliability.

The effects of three failure mechanisms are analyzed by RELIC: metal migration, hot electrons, and time dependent dielectric breakdown. Each of these mechanisms has a critical quantity which is indicative of the amount of stress being put on the device. The stress for each failure mechanism is a function of the voltages and currents on the circuit device under consideration; therefore a circuit simulator is necessary to calculate this information.

The RELIC simulator consists of three parts: a preprocessor, a modified circuit simulator, and a postprocessor. The preprocessor takes an input file which describes the circuit, and modifies it to include reliability test structures and "hidden" wear nodes. The modified circuit simulator then runs a simulation of the circuit and test structures and outputs plots of the voltages on the wear nodes. The postprocessor uses this wear information to predict when the various devices in the circuit will fail.

Microsystems Research Center Room 39-321 | Massachusetts Institute of Technology | Cambridge Massachusetts 02139 | Telephone (617) 253-8138

MASSACHUSETTS INSTITUTE OF TECHNOLOGY VLSI PUBLICATIONS

VLSI Memo No. 86-316
May 1986

COMPACTION WITH AUTOMATIC JOG INTRODUCTION

F. Miller Maley

Abstract

This thesis presents an algorithm for one-dimensional compaction of VLSI layouts. It differs from older methods in treating wires not as objects to be moved, but as constraints on the positions of other circuit components. These constraints are determined for each wiring layer using the theory of planar routing. Assuming that the wiring layers can be treated independently, the algorithm minimizes the width of a layout, automatically inserting as many jogs in wires as necessary. It runs in time $O(n^4)$ on input of size n. Several heuristics are suggested for improving the algorithm's practical performance.

The compaction algorithm takes as input a data structure called a sketch, which explicitly distinguishes between flexible components (wires) and rigid components (modules). The algorithm first finds constraints on the positions of modules that ensure enough space is left for wires. Next, it solves the system of constraints by a standard graph-theoretic technique, obtaining a placement for the modules. It then relies on a single-layer router to restore the wires to each circuit layer. An efficient single-layer router is already known; it is able to minimize the length of every wire, though not the number of jogs.

As given, the compaction algorithm applies only to a VLSI model that requires wires to run a rectilinear grid. This restriction is needed only because the theory of planar routing (and single-layer routers) has not yet been extended to other models. The compaction algorithm's correctness proof elucidates the assumptions on which the algorithm depends, so that the algorithm is easily generalized once the necessary theoretical machinery is in place.

Microsystems Research Center
Room 39-321

Massachusetts Institute of Technology

Cambridge Massachusetts 02139

Telephone (617) 253-8138

VLSI Memo No. 86-345
October 1986

WIRE-EFFICIENT VLSI MULTIPROCESSOR COMMUNICATION NETWORKS

William J. Dally

Abstract

VLSI communication networks are wire limited. The cost of a network is not a function of the number of switches required, but rather a function of the wiring density required to construct the network. This paper analyzes communication networks of varying dimension under the assumption of constant wire bisection. Expressions for the latency, average case throughput, and hot-spot throughput of these networks are derived. It is shown that low-dimensional networks (e.g., tori) have lower latency and higher hot-spot throughput than high-dimensional networks (e.g., binary n-cubes) with the same bisection width.

Microsystems Research Center
Room 39-321

Massachusetts Institute of Technology

Cambridge Massachusetts 02139

Telephone (617) 253-8138

VLSI Memo No. 86-346
October 1986

LAZY EVENT-DRIVEN SIMULATION

William J. Dally

Abstract

This paper describes a new simulation technique, lazy event-driven simulation. This technique reduces the computation required to simulate a circuit by avoiding the evaluation of modules whose outputs will later be ignored. The lazy simulation algorithm is presented along with performance measurements of a prototype lazy simulator.

Microsystems Research Center
Room 39-321

Massachusetts Institute of Technology

Cambridge Massachusetts 02139

Telephone (617) 253-8138

Acknowledgements

This work was supported in part by the Defense Advanced Research Projects Agency under contracts numbered N00014-80-C-0622 and N00014-85-K-0124.

Author Information

Dally: Artificial Intelligence Laboratory and Department of Electrical Engineering and Computer Science, MIT, Room NE43-417, Cambridge, MA 02139, (617) 253-6043.

Lazy Event-Driven Simulation

William J. Dally
Artificial Intelligence Laboratory
Massachusetts Institute of Technology

August 20, 1986

Abstract

This paper describes a new simulation technique, lazy event-driven simulation. This technique reduces the computation required to simulate a circuit by avoiding the evaluation of modules whose outputs will later be ignored. The lazy simulation algorithm is presented along with performance measurements of a prototype lazy simulator.

1 Introduction

Logic simulation is widely used to verify the behavior of logic circuits prior to their fabrication [2]. Simulation is critical for VLSI circuits since their internal nodes are inaccessible and once an error is discovered there is little possibility of repair. For large circuits, simulation is such a computationally intensive and thus costly process that special purpose simulation machines have been built [16].

Event-driven simulators reduce the amount of computation that must be performed by only evaluating gates with changing input values [15]. Typically only 2% to 10% of the gates in a circuit need to be evaluated in each simulation cycle of an event-driven simulator [3]. An event-driven simulator, however, performs superfluous evaluations when the output of a gate is ignored by subsequent stages of logic. This paper presents a new simulation technique, lazy simulation, that complements event-driven simulation by evaluating only those gates whose outputs are relevant.

Consider the circuit shown in Figure 1. As long as the lower input to the AND gate, E, remains zero, events on the inputs to the combinational logic block cannot affect the value of the output, OUT, and may be ignored. A conventional event-driven simulator will evaluate the logic block and the AND gate whenever one of the inputs (X, Y, or Z) changes. A lazy simulator avoids this unnecessary computation by ignoring all events on X, Y, or Z as long as E remains zero.

The remainder of this paper describes the mechanism of lazy simulation. Section 2 introduces the simulation model. The lazy simulation algorithm is presented in Section 3. An extension of the algorithm to handle circuits with delay is described in Section 4. Finally, the performance of an experimental lazy simulator is discussed in Section 5.

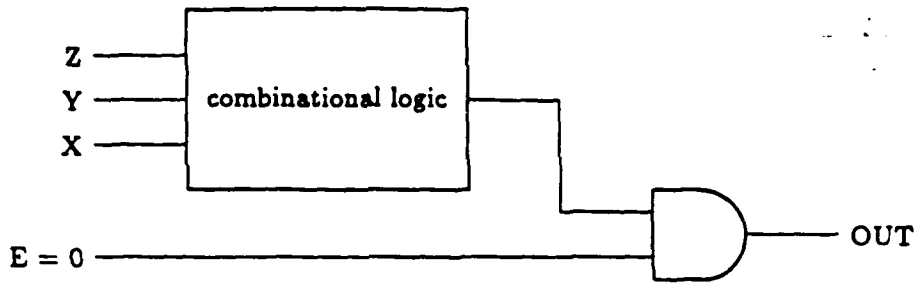


Figure 1: An Example Circuit

```
For each input vector
    Schedule input events on QUEUE
    While QUEUE is not empty
        Remove an event, EVENT = (BLOCK, VALUE), from QUEUE
        BLOCK.value ←VALUE
        For each input port, PORT, in BLOCK.fanouts
            ABLOCK ←PORT.block
            AVALUE ←ABLOCK.fun(ABLOCK.inputs)
            If AVALUE ≠ ABLOCK.value
                Schedule (ABLOCK, AVALUE) on QUEUE
```

Figure 2: Event-Driven Simulation Algorithm

2 Model

A circuit, $C = (B, P)$, consists of a set of *blocks*, $B = \{b_0, \ldots, b_n\}$, and a set of primary inputs, $P = \{p_0, \ldots, p_n\}$.

A block, $b \in B$, consists of a set of input ports, b.inputs, an output function, b.fun, and a fanout list, b.fanouts. The block containing input port i is denoted i.block. Each input port, i, of a block is connected to either a primary input, i.con $\in P$, or to a block, i.con $\in B$. A block b's fanout list b.fanouts contains all input ports connected to b. A block b has a value b.value that is computed by applying the block's output function, b.fun, to the values of the blocks connected to the elements of b.inputs.

When an input to the circuit changes, blocks are evaluated in any order until the circuit reaches a fixed-point. A circuit for which all input sequences have a unique sequence of fixed-points is said to have no *critical races* or *hazards* [6], [10].

3 Algorithm

The algorithm for normal event-driven simulation is shown in Figure 2. Blocks with inputs that have changed value are scheduled for evaluation on a queue. This has the effect

```
For each input vector
    Schedule input events on QUEUE
    While QUEUE is not empty
        Remove an event, EVENT = (BLOCK, VALUE), from QUEUE
        BLOCK.value ←VALUE
        If BLOCK is interesting
            For each input port, PORT, in BLOCK.fanouts
                If PORT is interesting
                    ABLOCK ←PORT.block
                    Compute interest of ABLOCK.inputs and propagate
                    AVALUE ←ABLOCK.fun(ABLOCK.inputs)
                    If AVALUE ≠ ABLOCK.value
                        Schedule (ABLOCK, AVALUE) on QUEUE
```

Figure 3: Lazy Event-Driven Simulation Algorithm

of evaluating the circuit breadth-first or in rank-order. Blocks connected to primary inputs (rank 1) are evaluated first. Blocks connected to a block in rank 1 (rank 2) are evaluated next and so on.

Lazy simulation extends normal event-driven simulation by keeping track of which blocks and input ports are *interesting* and only simulating events on interesting input ports. Whether a block or port is interesting or not is computed according to the following rules.

1. A block is always interesting if its output is a primary output of the circuit.
2. A block is always interesting if its output is a state variable.
3. A block is interesting if any of its fanouts are interesting.
4. An input port, i, is interesting if $b = i$.block is interesting and b.fun depends on i. For example, an input port of an interesting AND gate is only interesting if all other inputs are high.

The algorithm for lazy event-driven simulation is shown in Figure 3. It differs from the algorithm in Figure 2 in that blocks are not evaluated when events occur on uninteresting input ports. Also, the interest of a block's input ports is computed whenever a block is evaluated, and any changes in interest may be propagated to the block connected to the input port.

The code for propagating *interest* in ports and blocks is shown in Figure 4. If an interesting port, PORT, becomes uninteresting, the block connected to this port, PORT.block, becomes uninteresting if all other fanouts of this block are uninteresting. When an uninteresting port, PORT, becomes interesting, the block connected to this port must also be interesting. If this block was previously uninteresting it must be evaluated. This evaluation may in turn trigger the evaluation of other blocks.

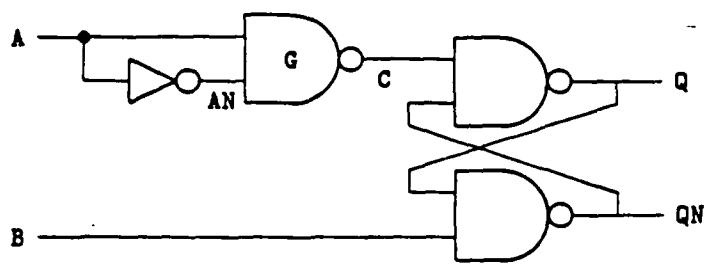
```
Become-Uninteresting(PORT)
    PORT.interest ←FALSE
    BLOCK ←PORT.con
    If all fanouts of BLOCK are uninteresting and BLOCK is not always interesting
        BLOCK.interest ←FALSE
        For each port, P, in BLOCK.inputs
            Become-Uninteresting(P)
END

Become-Interesting(PORT)
    PORT.interest ←TRUE
    BLOCK ←PORT.con
    If Not BLOCK.interest
        BLOCK.interest ←TRUE
        Compute interest of BLOCK.inputs and propagate
        BLOCK.value ←BLOCK.fun(BLOCK.inputs)
END
```

Figure 4: Propagating Interest

Because blocks are evaluated both by propagating events and by propagating interest, evaluation may take place in a different order than in a normal event-driven simulator. In circuits with hazards or critical races, changing the evaluation order may change the results of the simulation. Consider for example the circuit shown in Figure 5. Input A is changed from 0 to 1 while input B remains high. With normal event-driven simulation, evaluation is performed in rank order resulting in a low going pulse on the output of gate G. This static hazard will result in the simulated flip-flop going into oscillation. A lazy simulator, on the other hand, would initially consider the lower input to gate G and the inverter to be uninteresting. When gate G is evaluated following A being set to one, the inverter becomes interesting and is evaluated, giving an output of zero. Since one input of G is still zero, no glitch will appear on its output. If ternary simulation [5] is used, however, the glitch will be detected using either lazy or normal simulation. Section 4 presents an extension to the lazy simulation algorithm to handle circuits that depend on the delay of components and thus the order of their evaluation.

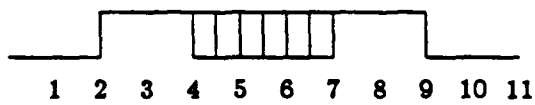
4 Delay

The operation of some logic circuits depends on the delay of the blocks composing the circuit. An example is the Queue circuit in Chapter 7 of Mead and Conway [11] that uses 3/2 rules. It requires that the delay through any three gates be at least as large as the delay through any two gates. To simulate such circuits we must modify our model to include the concept of delay. The output of a block must change its state a fixed period of simulated time after the input changed state. Simulating delays in circuits is also useful to verify




```
A  011111111 ...
AN 110000000 ...
C  110111111 ...
Q  000101010 ...
QN 111101010 ...
```

Figure 5: Circuit Containing a Hazard




```
{0 on (0,2), 1 on (2,4), 1 on (7,9), 0 on (9,11)}
```

Figure 6: Interval Representation of a Waveform

that the maximum delay through a circuit does not exceed some bound, although timing analyzers [9] [13] are usually more effective at performing this task.

We can extend the lazy simulation algorithm to handle circuits with delay, by considering values to be *waveforms*. A waveform is a list of interval-value pairs. Each pair represents the waveform being at a particular value for the specified time interval. An example of a waveform and its interval representation is shown in Figure 6.

Blocks operate on input waveforms to produce output waveforms as in the *T-algorithm* [7]. Unlike the T-algorithm however, we do not store or operate on the entire waveform at once. We only store that portion of the waveform necessary to allow a lazy simulator to produce the same result as a normal event-driven simulator, even though it evaluates blocks in a different order. A block that is interesting stores only its present value. When an input port becomes uninteresting it begins keeping a history of events. The history is kept only for the last d time units, where d is the maximum delay from the input port to a signal that is always interesting.

When an uninteresting input port, i, becomes interesting it must find its value at the present simulation time. If the block, $b = i.\text{block}$, is interesting, the port's history will contain this value. If b is uninteresting, however, the value of its inputs must be computed

Circuit	1-bit ALU	8-bit ALU	8-bit Mult	16-bit Arith.
Gates	17	136	640	976
Test Patterns	256	4096	4096	5120
Normal Evals	5509	350718	5679191	1626146
Lazy Evals	3893	267154	4171026	799371
Lazy / Normal	0.70	0.77	0.73	0.49

Table 1: Experimental Results for Normal and Lazy Simulation

at an earlier time, $T_d = T - b.\text{delay}$, where $b.\text{delay}$ is the delay of b. Block b is then simulated at time T_d, possibly causing the evaluation of other blocks at even earlier times if one of its input ports does not have the required value. This out-of-order evaluation is guaranteed to be consistent since the block was uninteresting. The roll-back is guaranteed to succeed because uninteresting input ports keep histories for the last d time units and the rollback will never exceed d.

5 Experimental Results and Analysis

A prototype lazy simulator has been implemented in LISP on a Symbolics 3600 computer. The simulator performs either normal or lazy event-driven simulation on gate-level circuits without delay. To measure the performance of the simulator, four test circuits were simulated using both normal and lazy simulation. The total number of evaluations performed during each simulation run are tabulated in Table 1.

The four circuits are a 1-bit ALU, an 8-bit ALU, an 8 × 8-bit multiplier, and a 16-bit arithmetic unit. The ALUs perform all 16 boolean functions of two variables as well as addition and subtraction. The multiplier is a simple combinational multiplier without recoding. A 16-bit ALU is combined with the multiplier to form the arithmetic unit. The output is selected by a multiplexor.

The results show that lazy simulation reduces the number of gate evaluations by 20 % to 50 %. The larger savings occur for circuits like the arithmetic unit that contain multiplexors. Since only one input of a multiplexor is interesting at a given time, these circuits contain many uninteresting nodes. Such circuits are quite common in digital systems.

The wall clock time per gate evaluation was almost identical for the two simulators, about 1ms. Thus, the cost of computing interest is comparable to the cost of normal event scheduling.

To date every circuit simulated on the prototype simulator has required fewer events for lazy simulation than for normal event-driven simulation. There is no guarantee that this will be the case however. A block is evaluated every time it becomes interesting. If the interest of a block changes several times while the inputs of the block remain stable, a lazy simulator would perform more evaluations than a normal simulator. This situation can be avoided by not allowing blocks to become uninteresting until they have processed at least one input event since they last became interesting. With this restriction it is easy to prove that in the worst case lazy simulation will require no more than twice as many

k	0	1	2	3	4	5	6	7
$P_o(k)$	1.00	0.80	0.64	0.51	0.41	0.33	0.26	0.21

Table 2: Probability of a Gate Being Interesting by Level, $F_i = F_o = 3$

evaluations as normal simulation. For most circuits, lazy simulation will require many fewer evaluations.

The order in which block inputs are evaluated is an important factor affecting the performance of a lazy simulator. The prototype simulator searches for the first signal in a dominating state (e.g., a zero input to an AND gate). Thus it is important that slowly changing control inputs be ordered before data inputs. For example, in a multiplexor the select inputs and their complements should be ordered before the data inputs.

If we assume that the values of the signals in a circuit are independent random variables we can estimate the fraction of nodes that will be interesting at a given time. While this assumption is certainly not true in practice (signal values are highly correlated) this approximation gives us a bound on activity, and a similar assumption has been shown to produce good results in grading tests for fault coverage [8]. Consider a circuit in which each gate has fan-in F_i and fan-out F_o, and each signal is equally likely to be high or low. Then the probability of the input of an AND or OR gate being interesting is

$$P_i(k) = 2^{-F_i} P_o(k) \left(1 + \frac{2^{F_i} - 1}{F_i}\right). \tag{1}$$

where $P_o(k)$ is the probability of the gate itself being interesting, and k is the distance from the gate to a primary output. P_o is a function of the distance from the gate to the output. Assuming that gates are in strict rank-order it is given by

$$P_o(k) = \begin{cases} 1 & \text{if } k = 0 \\ 1 - (1 - P_i(k-1))^{F_o} & \text{otherwise} \end{cases} \tag{2}$$

For the case where fan-in and fan-out are both three, the probability of a gate being interesting as a function of its level (minimum path-length to an output) is shown in Table 2. This data agrees to within about 10 % for the ALUs (average level $\approx$ 2), but severely underestimates the interest of gates in the multiplier (average level $\approx$ 4). This disagreement is due to the extensive use of exclusive-OR gates in the multiplier that propagate interest without attenuation back from the outputs.

6 Conclusion

Lazy simulation substantially reduces the amount of computation required to simulate digital circuits. By simulating only *interesting* blocks, lazy evaluation avoids computing the values that will subsequently be ignored. Only gates with changing inputs and interesting outputs are simulated.

There are many possible extensions to the work described here. The lazy simulation algorithm described here treats all state variables as interesting. A simulator could be constructed that computes state variables only when they are needed. It is difficult, however, to compute when a state variable is interesting. Simply propagating interest backwards results in cycling repeatedly around feedback paths. A more clever algorithm is required for propagating interest through feedback loops.

The algorithm developed here would apply equally well to concurrent fault simulation [14]. In a lazy fault simulator blocks may be interesting in some *machines* and uninteresting in other *machines*. Blocks would only need to be evaluated for the machines in which they are interesting.

Lazy algorithms can be applied to switch-level simulation [4]. Every node in a switch-level circuit contains state and thus would be interesting to an algorithm such as the one described here. Instead of applying lazy simulation to nodes at the logic level, however, we can apply the technique to transistors when paths are being traced to compute strengths. MOSSIM in fact has a mode in which it avoids propagating paths across transistors that it considers redundant.

The lazy simulation algorithm is an interesting hybrid of data-driven and demand-driven evaluation. Both of these techniques offer a flexibility in scheduling that can be used to exploit parallelism. This paper has only considered a sequential algorithm for lazy simulation. The technique, however, should be applicable to distributed simulators such as those described in [1] and [12].

References

[1] Ashok, V., Costello, R., and Sadayappan, P., *Distributed Discrete Event Simulation using Dataflow*, Ohio State University Technical Report OSU-CISRC-TR-85-8, February 1985.

[2] Breuer, M.A, and Parker, A.C., "Digital System Simulation: Current Status and Future Trends," *Proc. 18th Design Automation Conference*, 1981, pp. 269-275.

[3] Breuer, M.A., and Friedman, A.D., *Diagnosis and Reliable Design of Digital Systems*, Computer Science Press, 1976, p. 207.

[4] Bryant, R.E., "A Switch-Level Model and Simulator for MOS Digital Systems," *IEEE Transactions on Computers*, vol. C-33, no. 2, February 1984, pp. 160-177.

[5] Eichelberger, E.B., "Hazard Detection in Combinational and Sequential Switching Circuits," *IBM J. Research and Development*, vol. 9, no. 3, March 1965, pp. 90-99.

[6] Hill, F.J., and Peterson, G.R., *Introduction to Switching Theory and Logical Design*, Wiley, 1974.

[7] Ishiura, N., Yasuura, H., and Yajima, S., "Time First Evaluation for High-Speed Logic Simulation," *Proc. IEEE International Conf. on Computer-Aided Design, ICCAD-84*, November 1984, pp. 197-199.

[8] Jain, S.K., and Agrawal, V.D., "STAFAN: An Alternative to Fault Simulation," *Proc. 21st Design Automation Conference*, 1984, pp. 18-23.

[9] Jouppi, N.P., "TV: An nMOS Timing Analyzer," *Proc. Third Caltech Conference on VLSI,* R. Bryant, Ed., Computer Science Press, March 1983, pp. 71-86.

[10] Kohavi, Zvi, *Switching and Finite Automata Theory*, McGraw-Hill, 1970.

[11] Mead, C., and Conway, L., *Introduction to VLSI Systems,* Addison Wesley, 1980, pp. 259-260.

[12] Misra, J. "Distributed Discrete Event Simulation," *to appear in Computing Surveys.*

[13] Ousterhout, J.K., "Crystal, A Timing Analyzer for nMOS VLSI Circuits," *Proc. Third Caltech Conference on VLSI,* R. Bryant, Ed., Computer Science Press, March 1983, pp. 57-70.

[14] Ulrich, E., and Baker, T., "The Concurrent Simulation of Nearly Identical Digital Networks," *Computer,* vol. 7, no. 4, April 1974, pp. 39-44.

[15] Ulrich, E., and Baker, T., "Exclusive Simulation of Activity in Digital Networks," *CACM,* vol. 13 no. 2, February 1969, pp. 102-110.

[16] "ZyCAD LE-001 and LE-002 product description," ZyCAD, 1982.

VLSI Memo No. 86-320
June 1986

TIGHT BOUNDS FOR MINIMAX GRID MATCHING, WITH APPLICATIONS TO THE AVERAGE CASE ANALYSIS OF ALGORITHMS

Tom Leighton and Peter Shor

Abstract

The minimax grid matching problem is a fundamental combinatorial problem associated with the average case analysis of algorithms. The problem has arisen in a number of interesting and seemingly unrelated areas, including wafer-scale integration of systolic arrays, two-dimensional discrepancy problems, and testing pseudorandom number generators. However, the minimax grid matching problem is best known for its application to the maximum up-right matching problem. The maximum up-right matching problem was originally defined by Karp, Luby and Marchetti-Spaccamela in association with algorithms for 2-dimensional bin packing. More recently, the up-right matching problem has arisen in the average case analysis of on-line algorithms for 1-dimensional bin packing and dynamic allocation.

In this paper, we solve both the minimax grid matching problem and the maximum up-right matching problem. As a direct result, we obtain tight upper bounds on the average case behavior of the best algorithms known for 2-dimensional bin packing, 1-dimensional on-line packing and on-line dynamic allocation. The results also solve a long-open question in mathematical statistics.

Microsystems Research Center Room 39-321 | Massachusetts Institute of Technology | Cambridge Massachusetts 02139

Telephone (617) 253-8138

A Survey of Problems and Results for Channel Routing

(Extended Abstract)

Tom Leighton

Mathematics Department and

Laboratory for Computer Science

Massachusetts Institute of Technology

Cambridge, Massachusetts 02139

Abstract: This paper presents a brief survey of the known models, results, open questions and references for routing in a 2-sided channel with the specific goal of minimizing channel width.

This paper was supported by Air Force Contract AFOSR-86-0078, DARPA Contract N00014-80-C-0622 and an NSF Presidential Young Investigator Award with matching funds from Xerox and IBM.

1. Introduction

Channel routing plays a central role in the development of automated layout systems for integrated circuits. Many layout systems first place modules on a chip or circuit board and then wire together terminals on different modules that should be electrically connected. This wiring problem is often solved by heuristically partitioning the given space into rectangular channels and then assigning to each such channel a set of wires which are to pass through it. This solution reduces a "global" wiring problem to a set of disjoint (and hopefully easier) "local" channel routing subproblems. For this reason, channel routing problems have been intensively studied for over a decade, and numerous heuristics and approximation algorithms have been proposed.

In this paper, we consider a very specific class of channel routing problems with the following generic form. The *channel* consists of a rectilinear grid of *tracks* (or rows) and *columns*. Along the top and bottom tracks are numbers called *terminals*, and terminals with the same number form a *net*. A net with r terminals is called an *r-point net*. The smallest net is a *2-point* net. If $r > 2$, we have a *multipoint net*. The channel routing problem is to connect all the terminals in each net using horizontal and vertical wires which are routed along the underlying rectilinear grid. The goal is to complete the wiring using the minimum number of tracks; i.e., to minimise the *width* of the channel. No constraint is placed on the number of columns used at either end of the channel.

Wire segments are physically located on one of the one or more *layers*. Wire segments in different layers can be connected with *contact cuts*, which can be thought of as very short wire segments running through grid points perpendicular to the routing surface. We will not restrict the number of contact cuts, but no two wires can change layers in the same space without being connected.

A variety of models have been proposed for channel routing, with differences depending on the number of layers allowed and on the ways in which wires are allowed to interact. The simplest is *river routing*. In the river routing model, only one layer of interconnect is available. Since disjoint

wires cannot cross one another, a problem must be planar to be river routable. Unfortunately, this is often not the case in practice. Even worse, some simple problems like an N-net shift-left-by-1 require N tracks to river route. Hence, even though river routing problems can be routed optimally using a simple greedy algorithm, the model is sufficiently restrictive so as not to be relevant for general channel routing problems.

In this paper, we focus on models with two or more layers. Unfortunately, the problem of minimizing channel width is NP-complete for most of the models, so we are constrained to consider approximation algorithms and worst case bounds. Typically, the quality of the bounds and algorithms will be measured as a function of channel density, which is strongly correlated with optimal channel width. The *density* d of a channel routing problem is the maximum number of distinct nets crossing any vertical cut of the channel.

The paper is divided into five sections. Two-layer models for channel routing are discussed in Section 2. In particular, we describe the Manhattan, knock-knee, unit-vertical-overlap and unrestricted models. Models with three or more layers are discussed in Section 3. Included here is a discussion of multilayer knock-knee, restricted overlap and unrestricted models. The most important open questions concerning channel routing are discussed in Section 4. We conclude with some references in Section 5.

2. Models and Results for Two Layers

2.1. Manhattan Routing

The most common 2-layer model is the traditional *Manhattan routing model*. In the Manhattan model, horizontal wire segments are routed in one layer and vertical wire segments are routed in the other layer. Hence, wires can *cross* but cannot *overlap* (i.e., run on top of one another) for any distance. Note that when a wire changes direction, it must also change layers, which requires a contact cut at the corresponding grid point. Where a contact cut is used, no other electrically disjoint wire can pass through that grid point in either layer.

Szymanski [20] showed that optimal Manhattan routing is NP-complete, and with Yannakakis [21] showed that the problem remains NP-complete even for the special case of 2-point nets. Even worse, Brown and Rivest [5] showed that some simple problems with density 1 like the N-net 1-shift require $\Omega(\sqrt{N})$ tracks to route in the Manhattan model.

Motivated by the work in [5], Baker, Bhatt and Leighton [1] defined the notion of *channel flux* and showed that the flux of a channel is a lower bound on width. (Density was already known to be a trivial lower bound.) Roughly speaking, a channel has flux f if f is the largest integer for which some horizontal cut spanning $2f^2$ columns splits at least $2f^2 - f$ nets. The flux can be as large as $\sqrt{N}$ for a problem with N nets (e.g., a 1-shift) but is often constant in practice.

Most importantly, Baker, Bhatt and Leighton discovered a linear time approximation algorithm for Manhattan routing. For 2-point net problems, the algorithm uses just $d+O(f)$ tracks. For general problems with multipoint nets, the algorithm uses up to $2d + O(f)$ tracks.

2.2. Knock-Knee Routing

It is worth noting that the constraints of the Manhattan model can be reformulated so as to not require layer-per-direction routing. In particular, any two-layer routing for which wires do not overlap or share corners (e.g., no wire can bend on the top layer directly above a wire bending on the lower layer) can be assigned layers so that vertical segments are on one layer and horizontal segments are on the other layer.

When wires are allowed to share corners, however, we have the much different *knock-knee model* proposed by Rivest, Barats and Miller [18]. Although the knock-knee model seems close to the Manhattan model in terms of constraints, it is vastly superior when it comes to routing certain problems. For example, the N-net 1-shift can be routed using just 1 track in the knock-knee model whereas $\Omega(\sqrt{N})$ tracks are required in the Manhattan model. In fact, Rivest, Barats and Miller showed that every 2-point net problem can be routed using $2d - 1$ tracks in the knock-knee model. Hence, the flux lower bound disappears altogether. (Actually, there is a small bug in [18] which

has been noted and removed by many, including [14].) Density, of course, is still a lower bound. Somewhat surprisingly, Leighton [10] showed that the $2d-1$ bound is tight for every d in the worst case.

For multipoint net problems, the Rivest-Baratz-Miller algorithm is easily extended to route using $4d-1$ tracks. No better algorithm is known for general multipoint net problems, but Leighton [11] has discovered a $2d + O(\sqrt{d})$ algorithm for problems consisting solely of multipoint nets (i.e., no 2-point nets). In either case, no improved lower bounds are known.

2.3. Unit-Vertical-Overlap Routing

In the *unit-vertical-overlap model*, wires are allowed to overlap but only for unit segments at a time and only in the vertical direction. The unit-vertical-overlap model was first defined and studied by Leighton and Pinter [12] and Gao and Hambrusch [7]. Both groups discovered algorithms for this model that use only $d + O(d^{2/3})$ tracks to route 2-point net problems, and $2d + O(d^{2/3})$ tracks to route multipoint net problems. More recently, Berger, Brady, Brown and Leighton [2] improved the upper bounds to $d + O(\sqrt{d})$ and $2d + O(\sqrt{d})$, respectively, with a linear-time algorithm. Both bounds are a factor of two better than the corresponding knock-knee bounds, and the 2-point net bound, in particular, is very close to the general density lower bound.

The $d + O(\sqrt{d})$ bound for 2-point nets has been conjectured to be optimal in the worst case. In support of the conjecture, Berger, Brady, Brown and Leighton exhibit a large class of 2-point net problems which provably require $d + \Omega(\log d)$ tracks. No stronger lower bound is known for multipoint net problems.

The Berger-Brady-Brown-Leighton algorithm is also nice in the sense that it generalizes the Baker-Bhatt-Leighton Manhattan routing algorithm and the Rivest-Baratz-Miller knock-knee algorithm, which are very different. Development of algorithms that are tolerant to small changes in the model is worthwhile since they are likely to be more useful in practice, where small deviations from the norm are common.

2.4. Unrestricted Two-Layer Routing

In the *unrestricted model,* wires are allowed to overlap arbitrarily. Somewhat surprisingly, the algorithms for unrestricted routing are no better than those for unit-vertical-overlap routing, at least when measured in terms of worst-case channel width. The lower bounds are weaker, however. Half-density, not d, is the best universal lower bound and $d+1$ is the best worst-case bound known [2, 10].

3. Models and Results for More than Two Layers

3.1. Knock-Knee Models

Preparata and Lipski [15] defined the three-layer knock-knee model as a natural generalization of the two-layer knock-knee model. As in the two-layer case, wires are allowed to cross and share corners, but cannot overlap.

Once again, density is a lower bound for any problem in the three-layer knock-knee model. In a dramatic result, Preparata and Lipski proved that d tracks always suffice to route any 2-point net problem, which is precisely optimal. Later this was extended to $2d-1$ tracks for multipoint nets by Sarrafzadeh and Preparata [19], and to $\frac{3}{2}d$ for 3-point nets [16]. No nontrivial lower bounds are known for multipoint net problems.

Several researchers have also studied the less structured problem of knock-knee routing without worrying about layer assignment. This would be the equivalent of two or three layer knock-knee routing where layer changes are allowed in between grid points. Along these lines, Frank [6] obtained a variety of results, including those of Rivest, Baratz and Miller. In addition, Mehlhorn, Preparata and Sarrafzadeh [14] developed a linear time algorithm which reproduces the bounds of [15], [16], and [19]. Extending such results back to the more restrictive 3-layer knock-knee model is not always easy. In fact, Lipski [13] showed that it is NP-complete just to decide if it is possible to legally assign each wire segment to one of three layers. Brady and Brown [3], on the other hand, developed a fast algorithm that always legally assigns layers when four or more layers are allowed.

3.2. Restricted Overlap Models

Hambrusch [9] was the first to investigate a multilayer model in which restricted overlap is allowed. In particular, she showed how to route any 2-point net problem in $\frac{d}{k}+1$ tracks using L layers and k-fold overlap for $k \leq \lceil \frac{L}{2} \rceil - 2$. This is, of course, always within *one* track of optimal provided that at most k wires can overlap across any segment. For $\lceil \frac{L}{2} \rceil - 1 \leq k \leq \lceil \frac{L}{2} \rceil + 1$, Hambrusch described algorithms that differ from the lower bound of $\frac{d}{k}$ by a constant factor that tends to 1 as L gets large. For the special case when double overlap is allowed in 3 or 4 layers, the algorithm uses $\frac{3}{4}d+3$ or $\frac{2}{3}d+3$ tracks, respectively. No results were proved for multipoint net problems.

Brady and Brown [4] also considered multilayer models in which wires are allowed to overlap, but only on every Jth layer ($J \geq 2$). They showed how to route any multiterminal net problem on L layers using $\lceil (d+1)/\lceil L/J \rceil \rceil + 2$ tracks if $L \geq J+3$. For problems with a sufficient number of layers, this is within 3 tracks of the trivial lower bound $\lceil d/\lceil L/J \rceil \rceil$. For the special case of nonadjacent overlap ($J=2$) and 4 layers, they obtained a bound of $\lceil \frac{d+1}{2} \rceil + 4$ tracks, which is at most 5 tracks more than the simple lower bound of $\lceil \frac{d}{2} \rceil$. The upper bound can be improved to $\lceil \frac{d}{2} \rceil + 4$ tracks for 2-point net problems.

3.3 Unrestricted Overlap Models

For L-layer channels with unrestricted overlap, every problem requires at least $\frac{d}{L}$ tracks. Hambrusch [8] extended the lower bound to $\frac{d}{L-1}$ for a large class of 2-point net problems.

The first nontrivial upper bounds for this model are due to Hambrusch [9] who showed how to route 2-point net problems using $\lceil d/\lceil (L-1)/2 \rceil \rceil + 3$ tracks for $L \geq 5$. For 3 and 4 layers, the algorithm uses $\frac{2}{3}d+3$ and $\frac{3}{4}d+3$ tracks, respectively. These bounds were improved by Brady and Brown [4] who showed how to route any multipoint net problem using $\lceil (d+2)/\lfloor 2L/3 \rfloor \rceil + 5$ tracks for $L \geq 7$. Inferior results were obtained for smaller L.

Most recently, Berger, Brady, Brown and Leighton [2] obtained nearly optimal bounds for unrestricted multilayer routing. In particular, they showed how to route any 2-point net problem

using $\frac{d}{L-1} + O(\sqrt{d/L})$ tracks and any multipoint net problem with $\frac{d}{L-2} + O(\sqrt{d/L})$ tracks. The 2-point net bound is within an additive factor of $O(\sqrt{d/L})$ of the worst-case lower bound. The multipoint net bound is within a small constant factor of the general lower bound. All of the algorithms run in linear time.

4. Key Open Questions

The most important open question is whether or not multipoint net problems really require wider channels in 2-layer models than 2-point net problems. In all four of the 2-layer models described, the best upper bounds for multipoint net problems are a factor of two worse than the corresponding bounds for 2-point net problems. The work of Preparata and Sarrafzadeh [16] suggests that intermediate results for r-point net problems are possible, but no approach to the general problem has been uncovered.

The 2-point/multipoint net question can also be asked for $L > 2$ layers. In general, the best upper bounds for multipoint nets are a factor of $\frac{L-1}{L-2}$ worse than the corresponding 2-point net bounds.

It would be nice to determine exact worst case bounds for unrestricted channel routing in 2 or more layers, even for 2-point nets. Currently, the best universal upper bound differs from the best existential lower bound by an additive factor of $O(\sqrt{d/L})$ for 2-point net problems. My guess is that the upper bounds are fairly tight. In other words, I suspect that there are 2-point net problems that require $\frac{d}{L-1} + \Omega(\sqrt{d})$ tracks for constant $L \geq 2$.

Along the same lines, it would be interesting to know how unrestricted the wiring has to become before the worst-case lower bounds for unrestricted routing are achieved. For example, the best algorithm (in terms of worst-case performance) for unrestricted 2-layer routing produces the same worst-case bounds in the more restrictive unit-vertical-overlap model. Is it possible that unit-vertical-overlaps are sufficiently powerful to achieve the same worst-case performance as unrestricted routing? I suspect that this may be true.

Lastly, it is always worthwhile to develop algorithms which are guaranteed to perform close to optimal for any given problem, and not just for worst-case problems. For example, maybe there is a 2-layer knock-knee algorithm that always performs within a $(1+\epsilon)$ factor times optimal. Perhaps the elegant techniques recently developed by Raghavan [17] could be useful in developing such algorithms.

5. Selected References

1. Baker, B., S.N. Bhatt, and T. Leighton, "An Approximation Algorithm for Manhattan Routing," pp. 205–229 in *Advances in Computing Research 2 (VLSI Theory)*, ed. F.P. Preparata, JAI Press (1984).

2. Berger, B., M. Brady, D.J. Brown and F.T. Leighton, "Nearly Optimal Bounds and Algorithms for Multilayer Channel Routing," unpublished manuscript (1986).

3. Brady, M., and D.J. Brown, "VLSI Routing: Four Layers Suffice," pp. 245–257 in *Advances in Computing Research 2 (VLSI Theory)*, ed. F.P. Preparata, JAI Press (1984).

4. Brady, M., and D.J. Brown, "Optimal Multilayer Channel Routing with Overlap," *4th MIT Conf. on Advanced Research in VLSI*, MIT Press, pp. 281–298 (1986).

5. Brown, D.J., and R.L. Rivest, "New Lower Bounds for Channel Routing," *Proc. 1981 CMU Conf. on VLSI*, pp. 178–185 (1981).

6. Frank, A., "Disjoint Paths in a Rectilinear Grid," *Combinatorica*, Vol. 2, No. 4, pp. 361–371 (1982).

7. Gao, S., and S. Hambrusch, "Two-Layer Channel Routing with Vertical Unit-Length Overlap," to appear in *Algorithmica* (1986).

8. Hambrusch, S., "Using Overlap and Minimizing Contact Points in Channel Routing," *Proc. 21st Annual Allerton Conf. on Comm., Control, and Comp.*, pp. 256-257 (1983).

9. Hambrusch, S., "Channel Routing Algorithms for Overlap Models," *IEEE Trans. on Computer-*

Aided Design of Integrated Circuits and Systems CAD- 4(1) pp. 23–30 (Jan. 1985).

10. Leighton, F.T., "New Lower Bounds for Channel Routing," MIT VLSI Memo 82-71 (1981).

11. Leighton, F.T., 18.435 Class Notes (1985).

12. Leighton, F.T., and R.Y. Pinter, unpublished notes (1983).

13. Lipski, W., "On the Structure of Three-Layer Wireable Layouts," pp. 231–243 in *Advances in Computing Research 2 (VLSI Theory)*, ed. F. Preparata, JAI Press (1984).

14. Mehlhorn, K., F. Preparata and M. Sarrafzadeh, "Channel Routing in Knock-Knee Mode: Simplified Algorithms and Proofs," Univ. Saarbrucken Technical Report (Nov. 1984).

15. Preparata, F.P., and W. Lipski, "Optimal Three-Layer Channel Routing," *IEEE Trans. on Computers* C-33 pp. 427–437 (1984).

16. Preparata, F.P., and M. Sarrafzadeh, "Channel Routing of Bounded Degree Nets," pp. 189–203 in *VLSI Algorithms and Architectures*, ed. P. Bertolazzi and F. Luccio, North-Holland (1985).

17. Raghavan, P., "Probabilistic Construction of Deterministic Algorithms: Approximating Packing Integer Programs," to appear in the *27th IEEE Conf. on Foundations of Computer Science* (1986).

18. Rivest, R.L., A. Baratz, and G. Miller, "Provably Good Channel Routing Algorithms," *1981 CMU Conf. on VLSI Systems and Computations*, pp. 158–159 (Oct. 1981).

19. Sarrafzadeh, M., and F.P. Preparata, "Compact Channel Routing of Multiterminal Nets," *Annals of Discrete Math.* 109 (Apr. 1985).

20. Szymanski, T.G., "Dogleg Channel Routing is *NP*-Complete," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems* CAD-4(1) pp. 31–41 (Jan. 1985).

21. Szymanski, T.G. and M. Yannakakis, personal communication (1982).

A Hyperconcentrator Switch for Routing Bit-Serial Messages
(Extended Abstract)

Thomas H. Cormen
Charles E. Leiserson

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

Abstract

In highly parallel message routing networks, it is sometimes desirable to concentrate relatively few messages on many wires onto fewer wires. We have designed a VLSI chip for this purpose which is capable of concentrating bit-serial messages quickly. This hyperconcentrator switch has a highly regular layout using ratioed nMOS and takes advantage of the relatively fast performance of large fan-in NOR gates in this technology. A signal incurs exactly $2 \log_2 n$ gate delays through the switch, where n is the number of inputs to the circuit. The architecture generalizes to domino CMOS as well.

1 Introduction

The problem of concentrating relatively few communications on many input lines onto a lesser number of output lines must be solved in communication networks of all kinds. In many parallel computing systems, communications are packaged into messages which are routed among the processors. This paper presents a design for a VLSI implementation of a fast concentrator switch suitable for routing bit-serial messages in a parallel supercomputer.

An *n-by-m concentrator switch* has n input wires $X_1, X_2, \ldots, X_n$ and $m < n$ output wires $Y_1, Y_2, \ldots, Y_m$. The switch can establish m disjoint electrical paths from any set of m input wires to the m output wires. A concentrator switch always routes as many messages as possible. Specifically, whenever k out of the n input wires of an n-by-m concentrator switch carry messages, one of the following is true:

- If $k \leq m$, then an electrical path is established from each input wire which contains a message to an output wire.
- If $k > m$, then each output wire has an electrical path established from an input wire which contains a message.

When $k > m$, some messages cannot be successfully routed, in which case we say the switch is *congested*. Typical ways of handling unsuccessfully routed messages in a routing network are to buffer them, to misroute them, or to simply drop them and rely on a higher-level acknowledgment protocol to detect this situation and resend them. The switch design in this paper is compatible with any of these congestion control methods.

One way to create a concentrator switch is with a hyperconcentrator switch. An *n-by-n hyperconcentrator switch*[1] has n input wires $X_1, X_2, \ldots, X_n$ and n output wires $Y_1, Y_2, \ldots, Y_n$. The switch can establish disjoint electrical paths from any set of k input wires, for any $1 \leq k \leq n$, to the first k output wires $Y_1, Y_2, \ldots, Y_k$. In other words, we route the k messages to the first k output wires. We can make any n-by-m concentrator switch from an n-by-n hyperconcentrator switch by simply choosing the first m output wires of the hyperconcentrator switch, $Y_1, Y_2, \ldots, Y_m$, as the m output wires of the concentrator switch.

We can use a sorting network to implement a hyperconcentrator switch. The inputs to the sorting network are 1's and 0's, representing the presence or absence of messages on the input wires to the switch. The sorting of the 1's and 0's, with 1's before 0's, causes the k input messages to occupy the first k outputs.

Many sorting networks, such as Batcher's bitonic sort [6], employ the technique of recursive merging.

This research was supported in part by the Defense Advanced Research Projects Agency under Contract N00014-80-C-0622. Tom Cormen is supported in part by a National Science Foundation Fellowship. Charles Leiserson is supported in part by an NSF Presidential Young Investigator Award.

[1] The terminology is drawn from [11].

0190-3918/86/0000/0721 $01.00 © 1986 IEEE

Communication-Efficient Parallel Graph Algorithms

Charles E. Leiserson
Bruce M. Maggs

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

Abstract—Communication bandwidth is a resource ignored by most parallel random-access machine (PRAM) models. This paper shows that many graph problems can be solved in parallel, not only with polylogarithmic performance, but with efficient communication at each step of the computation. We measure the communication requirements of an algorithm in a model called the *distributed random-access machine* (DRAM), in which communication cost is measured in terms of the congestion of memory accesses across cuts of an underlying network. The algorithms are based on a communication-efficient variant of the tree contraction technique due to Miller and Reif.

Key Words: communication, fat-trees, graph theory, load factor, parallel algorithms, randomized algorithms, tree contraction, volume-universal networks.

This research was supported in part by the Defense Advanced Research Projects Agency under Contract N00014-80-C-0622. Charles Leiserson is supported in part by an NSF Presidential Young Investigator Award with matching funds provided by AT&T Bell Laboratories and Xerox Corporation. Bruce Maggs is supported in part by an NSF Fellowship.

VLSI Memo No. 86-343
October 1986

CRYPTOLOGY AND VLSI (a two-part dissertation):

I. Detecting and Exploiting Algebraic Weaknesses in Cryptosystems
II. Algorithms for Placing Modules on a Custom VLSI Chip

Alan Theodore Sherman

Abstract

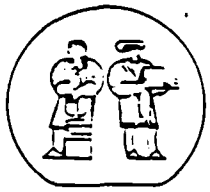
This dissertation describes two separate and independent investigations in cryptology and VLSI. Part I explores relationships between algebraic and security properties of cryptosystems, focusing on finite, deterministic cryptosystems whose encryption transformations form a group under functonal composition. Part II explores the problem of automatically placing modules on a custom VLSI chip, focusing on the placement heuristics used in the MIT PI (Placement and Interconnect) System.

Microsystems Research Center
Room 39-321

Massachusetts Institute of Technology

Cambridge Massachusetts 02139

Telephone (617) 253-8138



VLSI Memo No. 86-344
October 1986

A HIGH-PERFORMANCE VLSI QUATERNARY SERIAL MULTIPLIER

William J. Dally

Abstract

The QSM is a fast quaternary serial multiplier implemented in 4μ CMOS-SOS technology. This VLSI chip achieves high performance by using a quaternary representation of partial products, by using Domino-CMOS circuit design, and by recoding the multiplier. Encoding partial products as quaternary numbers enables this multiplier to use a novel pass-transistor circuit that performs two-bit addition four times faster than a conventional gate circuit. Prototype QSM chips on first silicon achieved a multiplication rate of 20M bits/sec. Simulation results indicate that a QSM designed in a state-of-the-art CMOS technology would achieve multiplication rates in excess of 200M bits/sec.

Microsystems Research Center
Room 39-321

Massachusetts Institute of Technology

Cambridge Massachusetts 02139

Telephone (617) 253-8138

Optimal Simulations of Tree Machines

(Preliminary Version)

Sandeep Bhatt
Dept. Computer Science
Yale University
New Haven, CT 06520

Fan Chung
Bell Comm. Research
Morristown, NJ 07960

Tom Leighton
Dept. Mathematics
M.I.T.
Cambridge, MA 02139

Arnold Rosenberg
Dept. Computer Science
U. Massachusetts
Amherst, MA 01003

Abstract—
Universal networks offer the advantage that they can execute programs written for simpler architectures without significant run-time overhead. In this paper we investigate simulations of tree machines; the fact that divide-and-conquer algorithms are programmed naturally on trees motivates our investigation.

Among various proposals for parallel computing the boolean hypercube has emerged as a particularly versatile network. It is well known that programs for multi-dimensional grid machines, for example, can be executed on a hypercube with no communications overhead by embedding the grid as a subgraph of the hypercube. Our first result is that a program for any tree machine can be executed on the hypercube with constant overhead. More precisely, every cycle of a synchronous binary tree can be simulated in $O(1)$ cycles on a hypercube, independent of the shape of the tree. The algorithm to embed the tree within the hypercube runs in polynomial time. We also give efficient simulations of arbitrary binary trees on the complete binary tree, the FFT and shuffle–exchange networks.

It is natural to ask if any sparse network can simulate every binary tree efficiently. Somewhat surprisingly, we construct a universal bounded–degree network on N nodes for which every N node binary tree is a spanning tree. In other words, every binary tree can be simulated on our universal network with no overhead. This improves previous bounds on the sizes of universal graphs for trees.

1 Introduction

A number of supercomputer architectures interconnecting hundreds or thousands of processors have been proposed in recent years. Prominent is the boolean hypercube, different versions of which have been built at Intel, N-cube, BBN, and Thinking Machines. The hypercube offers a rich interconnection topology with high communication bandwidth, low diameter, and a recursive structure naturally suited to divide-and-conquer applications. More importantly, the hypercube supports efficient routing algorithms and can therefore simulate any realistic parallel machine efficiently. Using Batcher's deterministic sorting scheme or Valiant's randomized message routing algorithm for instance, the hypercube can simulate a PRAM, and hence any realistic parallel machine with only a small polylogarithmic multiplicative increase in time. This universality property makes the hypercube extremely attractive for parallel computing.

Many parallel architectures can be simulated on the hypercube without the logarithmic increase in time. For example, every $2^{d_1} \times \cdots \times 2^{d_n}$ grid is a subgraph of the $2^{\sum d_i}$ node hypercube. Such multi-dimensional grids can therefore be simulated on a hypercube with no communications overhead. The importance of grid algorithms to scientific applications coupled with the capability of a hypercube to simulate grids of different aspect ratios has often been cited as an important consideration in building hypercubes. Johnsson [12] gives a survey of various efficient matrix algorithms on the hypercube.

What other networks can be simulated on a hypercube with little or no communications overhead? This question remains largely unexplored. In this paper we take the first step to investigate simulations of binary trees within the hypercube. Highly parallel divide-and-conquer algorithms can be conveniently programmed on an abstract binary tree machine, as can concurrent data structures [1, 9]. While the complete binary tree is suitable for a number of applications, there are instances when the divide-and-conquer tree is not complete. For example, in finite-element computations the tree generated by recursively decomposing a region into smaller subregions is, in general, neither complete nor binary. This paper considers "static" simulations only; we assume that the binary tree is fixed in size and shape and

0272-5428/86/0000/0274$01 00

does not evolve in time. We also assume that all nodes of the tree (and not just the leaves) are active simultaneously, as is the case when different computations are pipelined through a single fixed tree. Our main result is that the hypercube can simulate every binary tree with only a small constant factor overhead in communications cost. This improves results of Bhatt and Ipsen [4] who give a simulation with communications overhead $\log\log N + O(1)$.

For many years graph theorists have been interested in constructions of universal graphs for important families of graphs. Chung and Graham [5] briefly describe some of the early work in this area. The archetypical problem is: given a class Ψ of graphs on N nodes, construct a universal graph H on N nodes with the fewest edges necessary so that every graph G in Ψ is a subgraph of H. The subgraph property is attractive in the context of parallel computing because it implies no communication overhead in simulating any graph in Ψ.

The case when Ψ is the set of N node trees has received considerable attention. Following [6, 7], Chung and Graham [8] constructed a universal graph for trees with $O(N \log N)$ edges, which is optimal up to constant factors for trees with unbounded degree [5]. For the case when Ψ is the class of trees of bounded degree, we give a bounded-degree universal graph. Since our graph has bounded-degree, it has a linear number of edges, thus improving the previous bound for arbitrary trees. Friedman and Pippenger [10] have recently shown that an expanding graph with N nodes and $O(N)$ edges is universal for binary trees with αN nodes $(0 < \alpha < 1)$.

The remainder of this paper is organized as follows. Section 2 gives definitions and illustrates simple embeddings of binary trees within the hypercube. Section 3 sketches the combinatorial argument basic to our embedding technique of Section 4 in which we describe how to simulate any binary tree on the hypercube with constant communication overhead. Section 5 describes the new construction of a bounded-degree universal graph for trees. Section 6 concludes with a number of extensions and open questions.

2 Definitions

The problem of simulating one network by another is modeled as a *graph embedding* problem. An embedding $< \phi, \rho >$ of a graph $G = (V_G, E_G)$ into a graph $H = (V_H, E_H)$ is defined by an injective mapping from V_G to V_H, together with a mapping ρ that maps $(u, v) \in E_G$ onto a path $\rho(\phi(u), \phi(v))$ in H that connects $\phi(u)$ and $\phi(v)$. The *dilation* of the edge (u, v) under $< \phi, \rho >$ equals the length of the path $\rho(\phi(u), \phi(v))$ in H. The *dilation* of an embedding $< \phi, \rho >$ is the maximum dilation, over all edges in G, under $< \phi, \rho >$.

We measure the quality of an embedding with three cost functions — *expansion, dilation,* and *load-factor.* Following Rosenberg [13], define the *expansion* of an embedding $< \phi, \rho >$ of G into H to be the ratio of the size of V_H to the size of V_G. Intuitively, expansion measures processor utilization. The *load-factor* $\lambda(e)$ of an edge e in H is the number of paths that pass through e which are images of edges in G, i.e., $\lambda(e_H) = |\{e \in E_G : \rho(e) \text{ contains } e_H\}|$, and the load-factor λ of an embedding is defined to be the maximum load-factor over all edges in H.

Our model of synchronous parallel networks assumes that a processor (node of G or H) can communicate with each of its neighbors in one clock cycle, so that edges serve as bidirectional links. We restrict attention to simulations in which each cycle of G is simulated by a series of cycles of H, before the simulation of the next clock cycle of G is begun. For an embedding $< \phi, \rho >$ of G, each communication across an edge in $e \in E_G$ is effected by transmitting the message along the path $\rho(e)$ in H.

Suppose we are given an embedding of G in H with dilation d and load-factor λ. It should be clear that the time to simulate one cycle of G on H can be no less than the dilation d. Furthermore, if every node in G communicates with each of its neighbors in one cycle, then as many as λ messages will pass across some edge in H in the same direction, so that the simulation must take at least λ cycles. Similarly, each message can be delayed at most λ cycles in a single queue so that if d is the dilation then $d\lambda$ cycles are sufficient to simulate one cycle of G. Summarizing, we have:

Lemma 1. *Let λ be the load-factor, and d the dilation of an embedding of G in H. If T cycles of H suffice to simulate any cycle of G using this embedding, then $\max\{d, \lambda\} \leq T \leq d\lambda$.*

To illustrate our definitions with an example, consider the embedding of Figure 1 of a complete binary tree within the hypercube. The nodes of the tree are numbered inorder — each node of the tree is mapped to the node in the hypercube with the corresponding address. Each edge from a node to its left child is mapped to the corresponding hypercube edge between the images of the two nodes, while the edge between a node and its right child is mapped to the path from the right child to the left child, and from the left child to the parent. The expansion, $N/(N-1)$, is the minimum possible while the dilation equals 2. The load factor also equals 2, but there are no queueing delays and two cycles of the hypercube

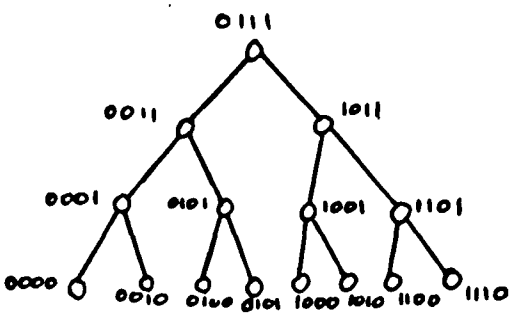


Figure 1: *The dilation 2 inorder embedding.*

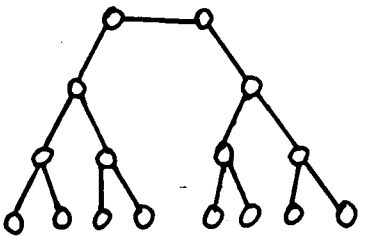


Figure 2: *A spanning tree S_4 of the hypercube.*

suffice to simulate one cycle of the complete binary tree.

Since an $N-1$ node complete binary tree is never a subgraph of the N node hypercube for $N \geq 8$ [14], any embedding of a large complete binary tree in the hypercube with expansion 1 must have dilation at least 2. In this sense, the inorder embedding of Figure 1 is optimal. There is, however, a much more efficient embedding. Bhatt and Ipsen [4] show that the N node tree $S_{\log N}$ of Figure 2 is a spanning tree of the N node hypercube. As a corollary, the $N-1$ node complete binary tree can be embedded in an N node hypercube with only one edge of dilation 2, and the with unit load-factor everywhere. In fact, this embedding is unique. Observe also that with expansion 2, the tree can be embedded with dilation 1.

Unfortunately, we do not know if arbitrarily structured binary trees can be efficiently embedded in such an elegant manner. We can use the well-known property that removing a single edge can separate a binary tree into two components each containing at least $\lfloor n/3 \rfloor$ nodes. By recursively embedding the split components within smaller hypercubes and translating one cube so that the nodes of the cut edge are adjacent in the new dimension, we obtain the following result.

Theorem 2. *Every N node binary tree can be embedded with unit-dilation in a hypercube with $O(N^{1.71})$ nodes.*

Although Theorem 2 gives a unit-dilation embedding, the expansion ($\approx N^{.71}$) is too large for the embedding to be useful in practice. In the next two sections we relax the unit-dilation requirement slightly, and show that every binary tree can be embedded with $O(1)$ dilation, expansion and load-factor.

3 The decomposition lemma

To embed an arbitrary N node binary tree T within the hypercube we proceed in two steps. In the first step T is decomposed and efficiently embedded within an N node *thistle tree*; an efficient embedding of the thistle tree within the hypercube in the second step induces an efficient embedding for T. This strategy is similar in spirit to the VLSI layout techniques of [2], with the thistle tree playing the role of the tree-of-meshes network. Combinatorial techniques developed previously for VLSI layout [2] apply in a straightforward way to the results in this section, and we will only sketch some of the proofs in this abstract. This section gives the combinatorial lemmas basic to our result; thistle trees and their embeddings are discussed in the next two sections.

Lemma 3. *Let T be any N node binary tree each of whose nodes is colored with one of k colors. Let n_i be the number of nodes of color i, $1 \leq i \leq k$, $\sum_i n_i = N$. By removing $k \log N$ or fewer edges, T can be bisected into two components of sizes $\lfloor N/2 \rfloor, \lceil N/2 \rceil$ such that, for each i, $1 \leq i \leq k$, each component has at least $\lfloor n_i/2 \rfloor$ nodes of color i.*

Lemma 4. *Every N node binary tree T can be mapped onto an N node complete binary tree C so that at most $6 \log \frac{N}{2^t} + 18$ nodes of T are mapped onto any one node of C at distance t from the root, and so that any two nodes adjacent in T are mapped to nodes at most distance 3 apart in C.*

Proof. The idea is to recursively bisect T, placing the successive sets of bisector nodes within successively lower levels of C, until T is decomposed into single nodes. For example, the nodes placed at the root of C bisect T into two subgraphs T_1 and T_2. Similarly, nodes mapped to the left child of the root bisect T_1 and nodes mapped onto the right child bisect T_2. In addition, at level i of C we map nodes of T (that have not already been mapped within levels $i-1, i-2$) that are adjacent to nodes mapped at level $i-3$ of C. This ensures that nodes adjacent in T will be mapped to nodes of C at most distance 3 apart.

To keep the number of nodes of T mapped to a level

i node in C within the required bounds, we use Lemma 4 with 3 colors. The following procedure describes how this is done.

1. *Step 1.* Initialize every node of T to color A, bisect T, and place the bisector nodes at the root (level 1).

2. *Step 2.* For each subgraph created in the previous step, recolor every node adjacent to the bisector in the previous step with color 0, and place a 2-color bisector for the subgraph at the corresponding level 2 node.

3. *Step 3.* For each subgraph created in the previous step, recolor every node of color A adjacent to the bisector in the previous step with color 1, and place a 3-color bisector for the subgraph at the corresponding level 3 node.

4. *Step t.* ($\log|T| \geq t \geq 4$). For each subgraph created in the previous step, place every node of color $t \pmod 2$ at the corresponding level t node, recolor every node of color A that is adjacent to one of color $t-1 \pmod 2$ with color $t \pmod 2$, and place a 3-color bisector for the remaining subgraph at the corresponding level t node.

If n_i is the maximum number of nodes mapped to a level i node of C, then we have $n_1 = \log N$, $n_2 = 2\log\frac{N}{2}$, $n_3 = 3\log\frac{N}{4}$, and because we use a 3-color bisector at each step, in general we have:

$$n_i \leq 3\log\tfrac{N}{2^i} + \tfrac{1}{2}n_{i-3},$$

from which the result follows. ∎

The decomposition is obtained in time polynomial in the size of T because, for fixed k, a k–color bisector for a tree can be found in polynomial time.

4 Embeddings in the hypercube

The decomposition obtained in the previous section motivates the definition of thistle trees. The thistle tree T_h of height h is obtained by starting with a complete binary tree of height h and, to each node at height i (leaves are at height 1), $1 \leq i \leq h$, attaching $i-1$ additional leaves. The thistle tree T_5 is shown in Figure 3. A simple calculation shows that the thistle tree $T_{\log N}$ of height $\log N$ has $2N - \log N - 2$ nodes.

The decomposition of Lemma 4 is invoked to embed an arbitrary N node binary tree within the thistle tree. By mapping the nodes of T that are mapped to the same internal node in Lemma 4 onto the corresponding thistle in the thistle tree (with at most $O(1)$ nodes of T at a single thistle) we can obtain an embedding with expansion 1 and dilation no greater than 5. In this embedding the thistles at the top levels may have multiple nodes of T embedded within them, but there is a corresponding deficit at thistles at the bottom of the thistle tree. By "pushing" the excess level-by-level down the tree, we can establish the following result.

Lemma 5. *Every N node binary tree T can be mapped onto a thistle tree with expansion 1 and $O(1)$ dilation.*

It remains to embed the N node thistle tree within the hypercube in an efficient manner. To this end, consider the inorder numbering of an $N/2$ node complete binary tree, as shown in Figure 4. It is not hard to see that each node u is within distance 1 (in the $N/2$ node hypercube) of every node along the rightmost path in the left subtree of u. Embed the thistle tree so that the centre of each thistle maps to the corresponding node in the complete binary tree, and each leaf maps to a distinct node in the rightmost path of the left subtree of the central node. Notice that the length of this path always equals the number of leaves hanging off the central node in the thistle. At this point each node of the hypercube has at most two thistle tree nodes mapped onto it. Now add another $N/2$ subcube and project each leaf of a thistle onto this "shadow" tree — this gives us an embedding of the thistle tree with expansion 1 and dilation 2. Together with Lemma 5, this guarantees an embedding of arbitrary T with expansion 1 and $O(1)$ dilation.

A more careful analysis in Lemma 5 shows that nodes u, v adjacent in T are mapped to thistles at most 6 levels apart. An interesting property of the inorder numbering is that the set of nodes obtained by picking a node and its descendants at distances $1, 2, \ldots, t$ induce a $t+1$ dimensional subcube (with one node missing), so that any two nodes at most $t-1$ levels apart are within distance $t+1$ in the hypercube. Therefore, by our earlier remark, the distance between the central nodes of the thistles for u and v are distance 8 apart in the hypercube. Since every central node is within distance 2 of its

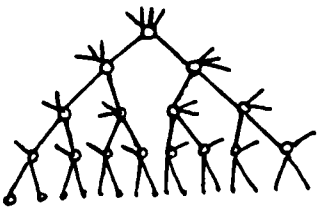


Figure 3: *The thistle tree T_5.*

leaves, the dilation of the overall embedding is at most 10. It is now straightforward to find paths in the hypercube between adjacent nodes of T so that the load factor is small. Summarizing, we have our main result of this section.

Theorem 6. *Every N node binary tree can be embedded in a hypercube with expansion 1, dilation 10 and $O(1)$ load-factor.*

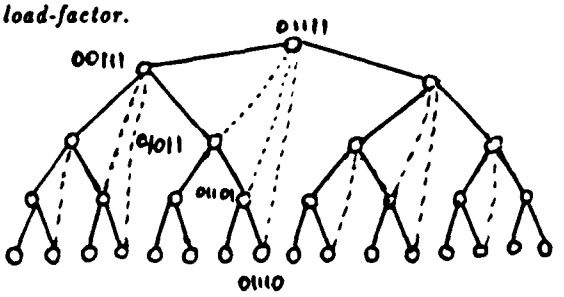


Figure 4: *Embedding the thistle tree within the complete binary tree.*

Together with Lemma 1, we have thus shown that every binary tree machine can be simulated with $O(1)$ communication overhead on a hypercube. The embedding can be computed in polynomial time because the bisection in Lemma 3, and consequently all other computations, can be computed in polynomial time.

5 An optimal universal graph

A graph H is said to be *universal* for a family of N node graphs if every graph in the family is a subgraph of H. The subgraph property is extremely strong (and attractive in applications) since it is equivalent to embeddings with unit dilation and load factor.

The problem of constructing N node universal graphs with fewest number of edges for all N node trees has received considerable attention. Following the work of [6, 7] Chung and Graham [8] constructed a universal graph for trees with $O(N \log N)$ edges. This bound is optimal, to within constant factors [5].

For binary trees, however, smaller universal graphs exist. Any N node binary tree can be embedded within an N node thistle tree with dilation 5. By connecting every pair of nodes that are at most distance 5 apart in the thitle tree, we obtain an N node graph with $O(N)$ edges that contains every N node binary tree as a spanning tree. However, the degree of the root is $O(\log N)$ so that although this universal graph is sparse, its nodes have unbounded degree.

There do however exist graphs with bounded-degree that are universal for all binary trees. This section gives the construction. First we need a few definitions.

Definition. A graph $G(V, E)$ is said to be *full* if for every $V' \subset V$, $|V'| \leq |V|/2$, the number of edges between V and V' is at least $|V'|$.

We observe in passing that there is a constant d such that for every m, there is an m node full graph with maximum degree d. In fact, any expander can be used for constructing full graphs.

The universal graph Γ on N nodes is obtained as follows. For simplicity, we will assume that $N = 2^a - 1$. Start the construction with a complete binary tree on N nodes. Then add edges so that the nodes at level k (a constant specified later) form a full graph on k nodes. Repeat this for nodes at levels $2k, 3k, \ldots$. Call the resulting graph Γ_0.

Next, add extra edges so that the nodes at levels $k, 2k, \ldots, \log N - s$ (k divides $\log N - s$ and s is a constant specified later) collectively form a full graph. Call the resulting graph Γ_1. Finally, insert an edge between any pair of nodes within distance t of each other, where t is a constant specified later. The resulting graph, denoted Γ, is our universal graph. Observe that the maximum degree of any node in Γ is no greater than $(2d+3)^t$ which, of course, is a constant because d and t are.

Of course, the construction given is primarily of theoretical interest because of the large constants. In the following subsections we establish the main result below.

Theorem 8. *Every N node binary tree is a spanning tree of Γ.*

5.1 Flow lemmas

The proof of Theorem 8 is somewhat involved, and requires a few combinatorial lemmas concerning full graphs and trees. The intuition captured in the following lemmas may be understood as follows. Suppose that we have mapped a subset of the nodes of a tree T within a graph G, and we next wish to map a node v of T onto a node of G in such a way that it remains "close to" its neighbors that have already been embedded. If there is no place readily available, we can still find a suitable place for v by "perturbing" the existing mapping slightly to make room for v. The "flow lemmas" establish conditions under which this can be done without dilating edges significantly.

Lemma 9. *Let G be a full graph with maximum degree d, and consider any assignment of packets to nodes of G such that every node of G is assigned at least $\lceil d/2 \rceil$*

packets. Then for any disjoint subsets S and T of nodes such that $|S| = |T|$, it is possible to redistribute the packets so that:

- *every packet either stays stationary or moves to a neighbor in G,*
- *the number of packets in each node in S decreases by 1,*
- *the number of packets in each node in T increases by 1, and*
- *the number of packets in each node in $V - (T \cup S)$ remains the same.*

Proof sketch: The lemma is proved with a simple max-flow/min-cut argument. Set up a flow problem with a supersource connected to each node in S and a supersink connected to each node in T. Assign unit capacity to each edge. Because G is full, there is a $0-1$ flow with value $|S|$ between the source and sink. The flow determines a 1-1 correspondence (along with edge-disjoint paths) from the nodes in S to the nodes in T. By moving one packet forward along each edge that has unit flow we can effect a reassignment of packets that satisfies the last three conditions of the lemma.

Since every node in the flow graph (with the supersource and supersink) has degree at most $d+1$, at most $\lfloor (d+1)/2 \rfloor = \lceil d/2 \rceil$ packets will be removed from any node of G during the reassignment process. Since every node of G initially has $\lceil d/2 \rceil$ packets, no packet need ever move more than one step. Hence the reassignment also satisfies the first condition.

Lemma 10. *Let G be an m node full graph with maximum degree d, and consider any assignment of packets to nodes of G so that node v_i has a_i packets, where $a_i \geq \lceil d/2 \rceil$ for $1 \leq i \leq m$. Then, for any set of numbers $\{a'_i \mid 1 \leq i \leq m\}$ for which $a'_i \geq \lceil d/2 \rceil$ for $1 \leq i \leq m$, it is possible to redistribute the packets so that*

- *every packet is reassigned to a node which is at distance at most $\max_{1 \leq i \leq m} |a_i - a'_i|$ from its original location in G, and*
- *the number of packets assigned to v_i changes from a_i to a'_i, for all $1 \leq i \leq m$.*

Proof sketch: Apply Lemma 9 $\max_{1 \leq i \leq m} |a_i - a'_i|$ times, each time decreasing the maximum value of $\max_{1 \leq i \leq m} |a_i - a'_i|$ by one, but otherwise preserving the hypothesis of the lemma.

5.2 Decompositons revisited

To establish Theorem 8 we use a decomposition strategy different from that in section 3. The following lemma is a simple extension of the $1/3 : 2/3$ separator theorem for binary trees and was observed previously in [3].

Lemma 11. *For every constant $p < 1/2$, there exists a constant q such that any m node two-colored binary forest with w nodes of color A can be partitioned into two sets by the removal of q edges so that each set has at least $\lfloor pm \rfloor$ nodes and at least $\lfloor pw \rfloor$ nodes of color A.*

We also require an additional, final lemma below.

Lemma 12. *Every N node binary tree T can be embedded within Γ_0 so that:*

- *every node in levels $0, k, 2k, \ldots, \log N - s$ of Γ_0 is assigned at least $\lceil d/2 \rceil$ and at most c_1 nodes of T, where c_1 is some constant,*
- *nodes of T are only assigned to nodes in levels $0, k, 2k, \ldots, \log N - s$ of Γ_0, and*
- *nodes adjacent in T are assigned to nodes in Γ_0 separated by distance at most c_2, for some constant c_2.*

Once Lemma 12 is established, it is easy to complete the proof of Theorem 8.

Proof of Theorem 8. First obtain the embedding of Lemma 12. Next, by Lemma 10 we can use the edges of $\Gamma_1 - \Gamma_0$ to reassign the nodes of T within Γ_1 so that:

- every node in levels $0, k, 2k, \ldots, \log N - s - k$ of Γ_1 is assigned $2^k - 1$ nodes of T,
- every node in level $\log N - s$ of Γ_1 is assigned $2^s - 1$ nodes of T, and
- nodes adjacent in T are assigned to nodes in Γ_1 separated by distance at most c_3, where $c_3 \leq c_2 + 2\max(|2^s - 1 - \lceil d/2 \rceil|, |2^k - 1 - c_1|)$.

At this point, we need only require that $s \geq k$ and that $2^k - 1 \geq \lceil d/2 \rceil$ so that the conditions of Lemma 10 are satisfied. Since k, s, d, c_1, and c_2 are all constants, we know that c_3 also is constant. We now reassign nodes one more time so that the mapping from T to Γ is 1-1 and onto. This is done by arbitrarily assigning the nodes of T on levels $0, k, 2k, \ldots, \log N - s$ of Γ_1 to their immediate descendants. Once this is done, the maximum distance in Γ_1 between any two nodes adjacent in T will be at most $c_3 + 2s$, which is constant. By setting $t = c_3 + 2s$ in

the construction of Γ, this will mean that T is a subgraph of Γ, thereby completing the proof. ■

Proof of Lemma 12. We follow an approach similar to that in section 3. However, since we are allowed to place only $O(1)$ nodes of T at any one node of Γ_0, we cannot afford to bisect the tree at each step because that may require placing placing $\Theta(\log N)$ nodes of T at the root of Γ_0. Therefore, instead of bisecting the tree at each step, we separate it into proportional size components using Lemma 11, and continually balance the sizes of components as the embedding proceeds towards lower levels of Γ_0.

Start by picking any $\lceil d/2 \rceil$ nodes of T and mapping them to the root (level 0) of Γ_0. Color *red* those nodes of T that are adjacent to one or more of the nodes placed at the root of Γ_0 (all nodes are initially colored white). Next, fix p to any constant greater than $1/3$ and use Lemma 11 to partition the (as yet unmapped) nodes of T into two sets, each with at least a fraction p of the total number of unmapped nodes, and each with at least a fraction p of the total number of red nodes (always rounded to the nearest integer, of course). One of the sets is distributed to the left subtree of the root of Γ_0 and the other set to the right subtree. By Lemma 11, no more than q edges connect nodes in the two sets.

No nodes of T will be assigned to the next $k-1$ levels of Γ_0, but we continue to partition T into smaller and smaller sets. In particular, we first color nodes in the *"left set" of T (those unmapped nodes of T distributed* to the left subtree of Γ_0 which are adjacent to nodes in the right set. We then use Lemma 11 to partition the left and right sets each into two smaller subsets, one for each grandchild of the root. Continue in this fashion, coloring nodes red as they become adjacent to nodes in the opposite set and splitting the forests (sets) into smaller forests until we have distributed a forest to each node on the kth level of Γ_0.

Although the nodes are split into roughly equal fractions $(p : 1-p)$ at each level, the sizes of forests at the kth level could vary substantially (in fact, anywhere between p^k and $(1-p)^k$). Therefore, at this stage we balance the sizes of the forests assigned to each node by redistributing forests among nodes at level k. To achieve this balance, first use Lemma 11 to partition each forest into $\lceil d/2 \rceil$ subforests (but do not distribute the subforests further down the tree). Next, partition each subforest whose size is greater than $1/p$ times the size of the smallest subforest. Observe that this does not affect the size of the smallest subforest.

We are now ready to apply Lemma 10, with each subforest represented as a packet. In particular, we use Lemma 10 to redistribute subforests on the level so that every node ends up with an equal number of subforests (*to within one*). We then map all the red nodes of T (i.e., those adjacent to nodes in different subforests) to the corresponding node of Γ_0 where the enclosing subforest is currently located, making sure to map at least $\lceil d/2 \rceil$ nodes of T to each node on level k in Γ_0 (if there are not enough red nodes, then use up some of the white nodes within the same subforest to make up the total. We will show later that there are always enough nodes overall so that this is possible).

After the mapping is completed for level k, recolor red all white nodes of T that are adjacent to nodes already mapped and henceforth regard the collection of subforests assembled at a single node of Γ_0 as a single forest. Next, repeat the process used on levels $1, 2, \ldots, k$ for levels $k+1, k+2, \ldots, 2k, \ldots, \log N - s$ where s is a constant still to be specified. At every kth level we rebalance and coalesce forests as on level k, and map all red nodes of T to the corresponding nodes of Γ_0. At level $\log N - s$ all the unmapped nodes of T (both red and white) are directly mapped to the corresponding node of Γ_0. Several details remain to be ironed out; however, it should be clear that nodes adjacent in T are mapped to nodes which are at most k levels apart in Γ_0.

The analysis needed to complete the proof is tedious, but not difficult. We start by letting r_{ik} be the maximum number of red nodes in any forest after all partitioning, balancing, coalescing, mapping and recoloring is done at level ik of Γ_0. Similarly, let z_{ik} be the maximum number *of nodes (both red and white) in the smallest forest* at level ik.

We will prove by induction that, for $ik \leq \log N - s$, $z_{ik} \geq 2^{-ik}N/6$,and
$r_{ik} \leq r' = 96(1+r)2^{-k}p^{-(k+\lceil \log \lceil d/2 \rceil \rceil + 1)}$

Observing that $r' \geq \lceil d/2 \rceil$, we note that both statements are trivially true for $i = 0$ and N sufficiently large. We next calculate bounds for r_{ik+k} and z_{ik+k} to proceed with the inductive step.

By Lemma 11, we know that

$$r_{ik+1} \leq (1-p)r_{ik} + 1 + q$$

and therefore, each forest at level $ik+k$ of Γ_0 has at most $(1-p)^k r_{ik} + (1+q)/p$ red nodes initially. The process of partitioning forests into subforests at level $ik+k$ cannot increase this value, but redistributing, coalescing and recoloring certainly can. To measure their effect, we need to bound the number of subforests that are located at any node following redistribution. This of course depends on the overall number of subforests, which in turn depends on the size of the smallest subforest.

The size of the smallest subforest at level ik is z_{ik}. Hence, the size of the smallest forest at level $ik+1$ is

at least $pz_{ik} - 1$. Applying the argument recursively, we find that the size of the smallest subforest at level $ik+k$ (after all the subdividing at this level is complete) is, for $p \leq 1/2$, at least

$$z_{ik}p^{k+\lceil \log\lceil d/2\rceil\rceil} - (1-p)^{-1} \geq p^{k+\lceil \log\lceil d/2\rceil\rceil}2^{-ik}N/6 - 2$$

For sufficiently large s (i.e., small enough i), this is at least $p^{k+\lceil \log\lceil d/2\rceil\rceil}2^{-ik}N/12$. Hence, the number of subforests at this level is no greater than $12 \times 2^{ik}p^{-(k+\lceil \log\lceil d/2\rceil\rceil)}$. The maximum number of subforests located at any node after balancing is therefore no greater than

$$1 + 12 \times 2^{-k}p^{-(k+\lceil \log\lceil d/2\rceil\rceil)} \leq 24 \times 2^{-k}p^{-(k+\lceil \log\lceil d/2\rceil\rceil)}.$$

Consequently, the maximum number of red nodes in any forest after rebalancing and coalescing is at most

$$\left((1-p)^k r_{ik} + (1+q)/p\right) 24 \times 2^{-k}p^{-(k+\lceil \log\lceil d/2\rceil\rceil)}.$$

Since mapping and recoloring can increase this at most by a factor of two, we have:

$$\begin{aligned} r_{ik+k} \quad \leq \quad & 48(1-p)^k r_{ik}2^{-k}p^{-(k+\lceil \log\lceil d/2\rceil\rceil)} + \\ & 48(1+q)2^{-k}p^{-(k+\lceil \log\lceil d/2\rceil\rceil)} \end{aligned}$$

By choosing $p > 1/3$ so that $(1-p)/2p < 1$, we have that for k sufficiently large (in terms of p and d):

$$r_{ik+k} \leq \tfrac{1}{2}r_{ik} + 48(1+q)2^{-k}p^{-(k+1+\lceil \log\lceil d/2\rceil\rceil)},$$

and thus,

$$r_{ik+k} \leq 96(1+q)2^{-k}p^{-(k+1+\lceil \log\lceil d/2\rceil\rceil)} = r',$$

as claimed.

We next complete the inductive step for z_{ik+k}. Since the largest and smallest subforests differ in size by at most a factor of $1/p$, the size of the smallest forest after balancing and coalescing is at least $\frac{p}{2}(N - r'2^{ik+k})2^{-(ik+k)}$, the one-half in front accounting for the fact that every node has the same number of packets to within one. After mapping and recoloring, the size of the smalest forest is

$$z_{ik+k} \geq \tfrac{p}{2}(N - r'2^{ik+k})2^{-(ik+k)} - r'.$$

With some additional calculations it can be checked that this is at least $2^{-(ik+k)}N/6$ for $p > 1/3$ and sufficiently large (but constant) s, thereby completing the proof of the claim.

By choosing s sufficiently large, we have shown that every node at levels $0, k, \ldots, \log N - s - k$ of Γ_0 is assigned at least $\lceil d/2 \rceil$ and at most r' nodes of T. Since s is constant, every node at level $\log N - s$ of Γ_0 is aassigned between $\lceil d/2 \rceil$ and c_1 nodes, where c_1 is some constant bigger than r'. Moreover, nodes of T are only assigned to nodes in levels $0, k, \ldots, \log N - s$ of Γ_0. Hence it remains only to show that nodes adjacent in T are assigned to nodes in Γ_0 separated by distance at most c_2, for some constant c_2. We already know that c_2 is at most k plus the distance subforests are allowed to move during the rebalancing step at every kth level. By Lemma 10, this distance is at most the largest number of subforests at any node before rebalancing. By the construction, this is at most some constant determined by p, d, k and s. ■

6 Extensions and conclusions

This paper gives the first non-trivial simulations of structures other than grids in the hypercube. The decomposition lemma (Lemma 4) for binary trees also provides optimal embeddings of binary trees within other networks. For example, we can show that every N node binary tree can be embedded within an N node complete binary tree with expansion 1 and dilation $O(\log\log N)$. This settles a conjecture of Hong, Mehlhorn and Rosenberg [11] who showed a lower bound of $\Omega(\log\log N)$ for this problem. By embedding a complete binary tree within the shuffle–exchange graph with expansion 1 and dilation 2, we obtain $O(\log\log N)$ dilation for arbitrary trees embedded within the shuffle–exchange graph. Similarly, we have recently shown that an N node complete binary tree can be embedded with constant expansion and dilation within the FFT network; once again, it follows that any N node binary tree can be embedded with constant expansion and $O(\log\log N)$ dilation within the FFT graph. We leave open the question whether these bounds are optimal to within constant factors.

All our results on embeddings within the hypercube extend to bounded degree graphs with $O(1)$ separators, and are not restricted to binary trees. While our simulations are optimal to within constant factors, there is much room for reducing the overhead in expansion and dilation further. It would be satisfying to discover simpler ways to embed binary trees in the hypercube. For example, we do not know of any binary tree that cannot be embedded in the hypercube either with expansion 1 and dilation 2 or with expansion 2 and dilation 1.

The construction of (unbounded degree) universal graphs with few edges for binary trees based on the decomposition lemma also extend to bounded-degree trees. We can also construct graphs with $O(N\log N)$ edges that are universal for bounded-degree planar graphs with N nodes. We leave open the question whether this bound is optimal to within constant factors.

An important problem concerns efficient simulations of planar graphs on the hypercube. To our knowledge, only the problem of embedding grids has been studied previously. Planar graphs arise in many scientific applications involving two-dimensional finite-element analysis. Similarly, little is known regarding lower bounds on embeddings. For example, we can prove that every N node graph with minimum bisection $\Omega(N)$ requires dilation $\Omega(\log N)$, the maximum possible. To our knowledge, no other lower bounds on embeddings in the hypercube are known.

In this paper we have only considered injective mappings of static structures. Depending upon the application, there are many interesting models. For example, if the leaves of a binary tree represent active processes and internal nodes are waiting processes, then only the leaves need be mapped to distinct nodes. In other applications, the tree may be much larger than the underlying network in which case we need to minimize dilation as well as maintain load balance. Embedding dynamically changing structures within the hypercube is important in many applications, and little is known in this area. Also interesting is the problem of *on-line* embeddings, in which the tree to be embedded grows one node at a time. We can show that any N node network for which every N node binary tree can be embedded on-line as a spanning tree must contain $\Omega(N^2)$ edges. In contrast, Friedman and Pippenger [10] show that if the size of the tree is small (a constant fraction of N) then $O(N)$ edges suffice.

Acknowledgements

Thanks to Michael Fischer and Lennart Johnsson for helpful discussions. Sandeep Bhatt was supported in part by NSF grant DCR 84-05478, Tom Leighton by grants AFOSR-86-0078, DARPA N00014-80-c-0622, and an NSF Presidential Young Investigator Award with matching funds from Xerox and IBM, and Arnold Rosenberg by NSF grant DMC 85-04308. The work was also supported in part by Bell Communications Research.

References

[1] J. Bentley and H.T. Kung, "A tree machine for searching problems," *Proc. Intl. Conf. Parallel Processing*, (1979).

[2] S.N. Bhatt and F.T. Leighton, "A framework for solving VLSI graph layout problems," *JCSS*, Vol. 28, No.2, (1984).

[3] S.N. Bhatt and C.E. Leiserson, "How to assemble tree machines," *Advances in Computing Research 2*, (F. Preparata editor) JAI press 1984.

[4] S.N. Bhatt and I. Ipsen, "Embedding trees in the hypercube," Yale University Research Report RR-443 (1985).

[5] F.R.K. Chung and R.L. Graham, "On universal graphs," *Proc. 2nd. Intl. Conf. on Combin. Math.*, (A. Gewirtz and L. Quintas, Eds.); *Ann. N.Y. Acad. Sci.* 136–140, (1979).

[6] F.R.K. Chung and R.L. Graham, "On graphs which contain all small trees," *J. Combin. Theory Ser. B* 24 14–23 (1978).

[7] F.R.K. Chung, R.L. Graham and N. Pippenger, "On graphs which contain all small trees II," *Proc. 1976 Hungarian Colloq. on Combinatorics*, 213–223, North Holland, (1978).

[8] F.R.K. Chung and R.L. Graham, "On universal graphs for spanning trees," *Proc. London Math. Soc.*, 27 203–211 (1983).

[9] W.J. Dally and C.L. Seitz, "The balanced cube: a concurrent data structure," Caltech Technical report 5174:TR:85 (1985).

[10] J. Friedman and N. Pippenger, "Expanding graphs contain all small trees," manuscript (1986).

[11] J.-W Hong, K. Mehlhorn and A.L. Rosenberg, "Cost trade-offs in graph embeddings, with applications," *J. ACM 30*, 709–728, (1983).

[12] L. Johnsson, "Basic linear algebra computations on hypercube architectures," Yale University Technical Report (1985).

[13] A.L. Rosenberg, "Issues in the study of graph embeddings," *Proc. Workshop on Graph-theoretic concepts in Computer Science*, (1980).

[14] Y. Saad and M. Schultz, "Topological properties of the hypercube," Yale University Technical Report (1985).

VLSI Memo No. 86-341
October 1986

RECENT RESULTS IN VLSI CAD AT MIT

Richard E. Zippel, Paul Penfield, Jr., Lance A. Glasser, Charles E. Leiserson, John L. Wyatt, Jr., F. Thomson Leighton, and Jonathan Allen

Abstract

Before the mid 1970's, integrated-circuit designs were comparatively simple, and the limits to complexity of an integrated circuit were imposed by semiconductor technology. Advances in semiconductor technology, however, have made it possible to put more on a single chip than can be designed easily and simply by a small group of people. As a result, for the past ten years or so, as the complexity of the designs has increased, the construction of integrated systems has been limited by both the fabrication technology and the cost of full custom design. As today's design crisis is solved, it is expected that the complexity limits will also include our inability to envision the types of systems to deploy in integrated form. The field will become system-limited, as well as technology-limited and design-limited.

MIT has responded to all this with a program of research and education in microsystems aimed at three principal areas: fabrication technology, design, and systems. This paper is devoted to a discussion of some of the recent results in the area of VLSI CAD. This work is carried out the context of research in other areas of VLSI, and so interactions are both possible and desirable.

This paper discusses only research carried out on the MIT campus. MIT Lincoln Laboratory has a strong program of research in VLSI which is separate.

Microsystems Research Center
Room 39-321

Massachusetts Institute of Technology

Cambridge Massachusetts 02139

Telephone (617) 253-8138

Acknowledgements

This work was supported in part by the Air Force Office of Scientific Research, under contracts F29620-81-C-0054, OSR-82-0326 and F49620-84-C-0004; in part by the Defense Advanced Research Projects Agency under contract N00014-80-C-0622; in part by a National Science Foundation grant ECS83-10941; in part by fellowships from the RCA Corporation, the Office of Naval Research, National Science Foundation, IBM Corporation, Bantrell Foundation and Analog Devices; and in part by the Harris Corporation.

Author Information

Zippel, Penfield, Glasser, Leiserson, Wyatt, Allen: Department of Electrical Engineering and Computer Science, MIT, Cambridge, MA 02139; Zippel, current address: Symbolics, 11 Cambridge Center, Cambridge, MA 02139; Penfield: Room 39-321, (617) 253-2506; Glasser: Room 36-880, (617) 253-4677; Leiserson: Room NE43-321, (617) 253-5833; Wyatt: Room 36-865, (617) 253-6718; Allen: Room 36-413, (617) 253-2509; Leighton: Department of Mathematics, MIT, Room NE43-332, (617) 253-5876.

Recent Results in VLSI CAD at MIT

Richard E. Zippel[1], Paul Penfield, Jr.[1], Lance A. Glasser[1], Charles E. Leiserson[1]
John L. Wyatt, Jr.[1], F. Thomson Leighton[2], and Jonathan Allen[1]

[1]Department of Electrical Engineering and Computer Science
[2]Department of Mathematics
Massachusetts Institute of Technology
Cambridge, MA 02139

1. Introduction

Before the mid 1970s, integrated-circuit designs were comparatively simple, and the limits to complexity of an integrated circuit were imposed by semiconductor technology. Advances in semiconductor technology, however, have made it possible to put more on a single chip than can be designed easily and simply by a small group of people. As a result, for the past ten years or so, as the complexity of the designs has increased, the construction of integrated systems has been limited by both the fabrication technology and the cost of full custom design. As today's design crisis is solved, it is expected that the complexity limits will also include our inability to envision the types of systems to deploy in integrated form. The field will become system-limited, as well as technology-limited and design-limited.

MIT has responded to all this with a program of research and education in microsystems aimed at three principal areas: fabrication technology, design, and systems. This paper is devoted to a discussion of some of the recent results in the area of VLSI CAD. This work is carried out in the context of research in other areas of VLSI, and so interactions are both possible and desirable.

This paper discusses only research carried out on the MIT campus. MIT Lincoln Laboratory has a strong program of research in VLSI which is separate.

2. VLSI Research Program

MIT recently opened a state-of-the-art VLSI fabrication facility on campus. It includes a fully equipped 6800 square-foot integrated-circuit laboratory, with about 2800 square feet of class 10 clean space and all the equipment necessary to process wafers from starting material through to final packaging and testing. Its purpose is to provide a stable, base-line process to support research in processing and process control, and to fabricate systems that require unusual process steps. There is also a 3600 square-foot class 100 lab dedicated to novel process technology development, and a 2000 square-foot Submicron Structures Laboratory, including 1000 square feet of class 10 space.

It is convenient to consider the MIT VLSI research program as being composed of seven interrelated parts: submicron technology, electronic materials, semiconductor processing, semiconductor devices, VLSI CAD, VLSI systems, and VLSI theory. The total program is roughly half devoted to the "electron-oriented" activities, and half to the "bit-oriented."

Much of the research in semiconductor processing and devices could not even be attempted without the new facilities mentioned above. This work includes a program, sponsored by the Semiconductor Research Corporation, on process technology for mixed analog and digital circuits. The research vehicle is a high-accuracy, high-speed A/D converter. It also includes a plan to develop software control systems for IC fabrication facilities. The purpose of the proposed Computer-Aided Fabrication system is to improve flexibility of fabrication, repeatability of processes and experiments, portability of processes, yield and efficiency of process development, thereby reducing overall cost and latency time in IC fabrication. These benefits arise from more complete documentation, elimination of paper in the clean environment, avoidance of human errors, optimal scheduling, environmental monitoring, inventory management, cost accounting, direct equipment drive, and convenient process-development and

process-analysis environments—in short by management of all the data associated with VLSI fabrication. This CAF work is part of the VLSI program and it is also part of a broader program of research in manufacturing.

At the other end of the spectrum, there is interest at MIT in highly parallel architectures, since such architectures match the capabilities of VLSI to provide massive amounts of logic. This work includes the development of dataflow machines, and particularly the languages and software systems necessary to make effective use of highly parallel systems, and of algorithms that are particularly well suited for such an environment.

Although most of the VLSI research can be conveniently classified as belonging to one of the seven areas mentioned above, some of the more interesting projects seem to defy such categorization and are truly interdisciplinary. One of the best examples is the development of a special content-addressable memory whose architecture has been designed to be usable in a wide variety of different applications [25, 30]. Making full use of this architecture is the subject of continuing research. To achieve the desired density and performance goals, novel circuits and a low-resistance interconnect technique have been developed. This effort is funded by a consortium of eight industrial firms, and involves three faculty members from diverse areas (L. R. Reif, processing; C. G. Sodini, circuits; and R. E. Zippel, architecture).

A paper this short cannot discuss research results from all these areas; instead, the focus is on recent results in VLSI CAD.

3. Recent results in VLSI CAD

Nine recent results or projects in VLSI CAD are discussed below. These include particular design tools, and also approaches that support the development of new tools.

3.1 CAD Tool Frame

SCHEMA [2] is an integrated design system for electronic designs being developed by a group led by Richard Zippel. Also involved in SCHEMA's development is a group of researchers at Harris Corp. being led by George Clark. Building VLSI systems is very complex undertaking. The complexity of the task is manifest in two fashions. First, the designs themselves are quite complex and the designers need help managing the complexity of the design. Second, the design tools themselves are becoming more and more complex. SCHEMA has been developed to address both of these types of complexity.

SCHEMA addresses the design complexity issue in three ways. First, SCHEMA provides a complete design structure in which the designer can store schematics, layouts, simulation results and all other artifacts of the design. Thus it captures the *complete* design. In order to understand the impact of different design decisions, the designer needs to examine the design from different perspectives. SCHEMA allows the designer to walk around the design laterally (through the schematics, layouts and simulation results) or vertically (from block diagram through logic and circuit schematics). Furthermore, SCHEMA maintains consistency between the different viewpoints of the design, if possible, and marks the different components incompatible if not.

The second approach SCHEMA uses to simplify large designs is to encourage the development of a library of *simple tools*. Simple tools perform a single, simple task to help the designer. Some examples are a *delay estimator* that calculates the delay between two nodes in a circuit or a *power estimator* that computes the average power dissipated by a set of devices. This information can be easily derived with a simulator or calculated by hand, but in either case requires enough effort on the designer's part that it is not often done. The purpose of the simple tools is to make this information available to the designer whenever and wherever it is needed so that design decisions can be made in a confident fashion. Another simple tool would convert a boolean equation into a pull-down or pull-up structure. Again an easy operation for a designer but somewhat time consuming. None of these tools is a major CAD advance in and of themselves but it is our belief that cumulatively they have a major impact on the designers productivity.

Finally, the big tools: simulators, routers and compactors, almost always have several variants. Simulations can be done using a switch level simulator, a logic simulator or a transient simulator. Routing can be done using any of a number of channel or switch box routers. Each of these techniques is appropriate for a different segment of a design. They should be built to be interchangeable, so that the designer can choose which one to use at different points in the design. The best example of this

type of CAD tool currently in use is a mixed mode simulator, though the decision process is controlled by the program not the designer.

To support these types of tools, the components of SCHEMA must be more thoroughly integrated than is of true most other CAD environments. This is accomplished by having the entire design and most design tools coexist in a common address/namespace. To encourage interchangeability in the software tools and to minimize the effort required to build CAD tools, a "Layered Language" software organization is used. Finally, a fairly complete and uniform toolbox of user interface tools is provided. Thus it is relatively rare that a tool designer need be concerned with developing a user interface to his or her tools.

3.1.1 Software Organization

The most common approach used in building a software system is to use a *structured design* methodology. With this methodology, a specification for the problem to be solved is given. Using this specification, the problem is decomposed into sub-problems, which are solved independently to encourage modularity. This approach is then applied recursively to each of the sub-problems. This leads to a tree-like decomposition of the problem into subproblems. This modular resolution of the problem makes it relatively easy to measure progress on the orignal goal and provides accountability for faults, important issues in developing large software systems.

We feel that this approach is weak for a number of reasons. First and most important, invariably the problem posed is not the problem that should be solved. Either the problem's specification is incomplete or inaccurate, or by the time the system is built the needs will have changed and a slightly different system is required. Second, the modularity imposed by structured design leads to duplication of effort as each team builds the tools necessary to solve their problem. If they choose to share modules with groups working on other subproblems, then hidden dependencies will appear among the modules sharing code, reducing their modularity. Finally, the accountability benefits of structured design are really illusory. Hard problems in the resulting system are more often due to poor decomposition of the problem; thus the responsibility for the problems is collective.

In SCHEMA we have chosen to use what we call a "layered language" approach to building systems. To solve a problem using this approach, one generalizes the problem to be solved into a class of related problems and then develops a "mini-language" for solving problems in this class. This new language has two components: (1) new data types and operations on the data types and (2) new control structures (abstractions). Collectively these tools should make the solution of problems from the original class easy. The resulting system consists of many language layers, each providing narrower, semantically richer mechanisms for describing the solution to a problem.

For instance, instead of building a single schematic entry system in SCHEMA, we have chosen to construct a language that makes it easy to construct graphics editing systems. This language includes spatial management data structures, icons and connectivity structures. New control structures exist for tracking the mouse, moving objects and synchronizing screen updates. This language is then used not only to build the schematic entry system, but also is the basis for the layout and waveform entry systems also.

The new data structures are often built using the Lisp Machine's "flavor system" and the new operations using "flavor methods" [18]. The multiple inheritance and sophisticated method combination mechanisms allow for greater sharing than many other approaches. New control structures are built using macros and functional arguments. A more detailed paper on this approach is in preparation.

3.2 Optimal Sizing of Transistors

Power consumption and signal delay are crucial to the design of high-performance VLSI circuits. Mark Matson and Lance Glasser have developed CAD tools for the modeling and optimization of digital MOS designs [15]. The tools determine the transistor sizes that minimize circuit power consumption subject to constraint on signal path delays. If a signal path constraint is unachievable then the tool returns the fastest possible design. Computational efficiency is obtained through macromodeling techniques and a specialized optimization algorithm. The macromodels are based on device equations and encapsulate logic gate behavior in a set of simple yet accurate formulas. The macromodels are valid for complex MOS gates with the limited

use of pass transistors. Typical accuracies are in the 5–10% range. The optimization algorithm exploits properties of the digital MOS domain to convert the primal optimization problem into a dual form which is much easier to solve. It handles the case of multiple constraint paths and transistor size constraints. Not all constraints need be active. The result of this research is a pair of CAD tools that can optimize a circuit in roughly the amount of time needed to perform a transistor level simulation of the circuit.

3.3 Reliability Simulation

RELIC [3, 4] is a reliability simulator developed to analyze stress and wear in MOS VLSI chips. Unlike its predecessors, RELIC is not designed around any particular failure phenomena or model but around the idea of reliability simulation in general. RELIC is the first reliability simulator to encompass several different failure mechanisms. RELIC is designed to be used in the circuit design phase to identify devices which are under excessive stress and to determine the worst-case reliability of the circuit. It is also helpful in comparing the median time to fail (MTTF) of different circuits and in determining whether or not a part can be effectively screened by accelerated testing.

RELIC uses a simple methodology for analyzing the stress caused by many different failure phenomena. A device which is stressed over time accumulates wear. The probability of device failure at time t depends on how much wear the device has collected by time t and the critical value of wear for that failure phenomenon or for that circuit. This critical value of wear might be the amount of trapped charge a gate oxide can take before undergoing TDDB; it can also be the threshold shift of a transistor under hot electron stress which causes the circuit to malfunction.

RELIC provides a number of features to aid the user in performing reliability analyses. A circuit device may be analyzed for any number of failure mechanisms, and multiple tests may be run with variations of model parameters. The circuit designer has the option of analyzing the entire circuit for reliability hazards, or concentrating on a few critical devices. RELIC employs a circuit simulator so that the voltages and currents used in stress calculations are worst-case operating waveforms and not just the maximum voltages and currents. This feature will become increasingly important as devices scale to the submicron regime.

The present implementation of RELIC includes three failure phenomena: metal migration, hot electron trapping, and time dependent dielectric breakdown. The metal migration model predicts and includes the effects of thermal transients in the wire. RELIC has been used to analyze several circuits. The most illustrative example is a bootstrapped superbuffer. The simulator successfully determined that one of the transistors in the circuit was being stressed at a rapid rate by both hot electrons and time dependent dielectric breakdown. RELIC also verified that a redesigned version of the superbuffer had a lower wear rate and hence larger MTTF.

3.4 Waveform Bounding

The purpose of this project is to develop theoretical foundations and practical algorithms for a new approach to fast timing analysis and simulation of monolithic digital VLSI circuits. The method is based on easily computed bounds for transient node voltages and signal propagation delays. It is intended as an alternative to the two methods that are currently standard practice: "exact" numerical solution and approximate delay formulas. It is an attractive alternative in many cases because exact numerical solution requires prohibitive amounts of computer time for VLSI circuits, and approximate delay formulas can result in large uncontrolled errors for some practical circuits.

The starting point for this work is the original paper by Rubinstein et. al. [22]. That paper provided computationally convenient upper and lower bounds for the step response of linear RC tree networks, which are useful as network models for the branching interconnect lines in MOS IC's. We have extended the usefulness of these results by generalising them to include interconnect networks with closed loops [27], such as the gate electrodes for large driver transistors. And we have proved theorems on the dynamics of RC networks that reduce the region of uncertainty in the original bounds. Tighter bounds have been achieved for all interconnect networks by exploiting slew rate limits on the node voltages, and further improvement was obtained for tree networks by using the spatial convexity of interconnect voltage during transients in a novel way [29,23]. Both these bound improvements have been successfully incorporated in a commercial CAD system for MOS standard cell design [24].

A project currently underway aims to extend the original work, intended for MOS applications, to include appropriate models for bipolar circuits [19]. The base-emitter junction in, e.g., ECL logic, provides a d.c. path to ground, not present in MOS, that invalidates the original bound formulas. The entire effort has been unified by the discovery [28] that the mathematical foundation of waveform bounding resides in the properties of linear dynamical systems governed by a special class of matrices, called "M-matrices."

A more broadly focussed and ambitious project is underway to develop a method for iterative bound relaxation, which will allow the user to flexibly trade off tightness of bounds for computer time as required at different stages in the design [31].

3.5 Network Extraction

A network extractor is a program which accepts an IC layout as input, and produces from it a machine-readable circuit representation, with sufficient information to perform detailed circuit simulation. The network extractor must be told about the fabrication technology and the meaning of the various layers; typically this is done through a "technology file" which is also treated as input. The output is typically in a form directly usable by a circuit simulator such as SPICE, but can also be used for layout verification, circuit schematic drawing, and electrical rule checking.

The network extractor EXCL has been developed by MIT by Steven P. McCormick and Jonathan Allen [16, 17]. The program has general extraction algorithms capable of accurate computations of interconnect resistance, inter-nodal capacitance, ground capacitance, and transistor size. However, where possible the general algorithms are replaced with simpler, faster techniques.

First, non-manhattan geometry is converted to staircase orthogonal geometry, since EXCL does not deal with arbitrary angles. Then comes a geometric decomposition phase, during which intersecting rectangles are grouped together, using technology-dependent intersection rules that are user-definable. Next is the extraction phase. Each transistor is transformed into its lumped equivalent. Each interconnect is also given a lumped representation. Separate algorithms for extracting capacitances and resistances are used. Again the extraction phase is user-controllable through technology-dependent extraction descriptions. Finally, the result is formatted for the desired use (logic simulation, timing simulation, circuit simulation, graphical presentation, etc.).

To extract interconnect resistance, EXCL uses three techniques. The most general technique works for arbitrary shapes, and solves Laplace's equation in two dimensions. The numerical techniques used are Gaussian elimination and successive over-relaxation. The second technique is valid for long straight wires with right-angle corners. This uses the standard formulas involving length and width, and standard corner corrections. The third technique uses a table lookup to handle commonly encountered geometry, such as tees, crosses, or vias.

To extract capacitance in the most general case is quite difficult. EXCL finds ground capacitance through calculations of areas and perimeters in a standard way. Coupling (internode) capacitance is calculated when EXCL judges that it will be significant. Three special techniques are used. For overlapping areas, the parallel-plate formula is used, with fringe correction. Parallel runs in the same or different layer are handled with a capacitance proportional to length, the constant of proportionality being a precomputed function of layer, separation, and technology. The third technique uses a lookup library to handle frequently occurring shapes. Besides these three special techniques, EXCL uses a Green's theorem approach for arbitrary geometry, if this is necessary.

3.6 Regular Structure Generation

A regular structure generator is a program which creates a VLSI layout of circuits which are repetitions of smaller circuits. Examples are n-bit adders which might be implemented as n full adders placed adjacent and wired together. Other regular structures include systolic arrays, multipliers, PLAs, and register files.

A regular structure generator has been developed at MIT by Cyrus S. Bamji and Jonathan Allen [1]. This program does much more than merely place leaf cells adjacent to each other, since in practice regular structures always have special edge or end conditions, and for optimum performance the leaf cells should be personalized according to the size of the array. For example, in an n-bit adder, the LSB can be a half adder rather than a full adder because there is no carry in.

The RSG uses previously defined cells to hierarchically build larger cells. It accepts a library of leaf cells, a parameter file, and a design file, and produces a circuit layout. The design file is a parameterized, procedural description of the architecture. The parameter file contains the parameters for the particular design desired. Hierarchy is used to enable the RSG to deal with macrocells, which are built up from subcells using legal interfaces. RSG can effectively deal with reflections and rotations of layouts. The legal ways of placing, orienting, and connecting cells are represented in an interface table.

The RSG operates as follows. First, a sample layout is read. From this, the RSG determines the leaf cells and some of the legal interfaces. Then new cells are created by building a connectivity graph for the new cell, and the converting this into a layout. The result is a new cell. If this new cell is to be itself used in a larger cell, new interfaces must be created. This process is repeated as necessary to hierarchically build up the final layout.

The RSG is written in the language CLU and consists of approximately 6000 lines of source code. It has been used to create a variety of layouts.

The previous results have been proven feasible by implementation in the form of CAD programs, either within MIT or outside. The results mentioned in the remaining sections have not yet been incorporated into working code.

3.7 Retiming

The goal of VLSI design automation is to speed the design of a system without sacrificing the quality of the implementation. A common means of achieving this goal is through the use of optimization tools that improve the quality of a quickly designed circuit. A group of researchers led by Charles Leiserson have developed a class of optimization techniques for clocked circuits called *retiming*, that relocate registers so as to reduce combinational rippling [10, 11, 12]. Unlike pipelining, these techniques do not increase circuit latency.

These techniques are most appropriate for circuits with a relatively large number of clocked stages such as signal processing systems. This sort of problem can be represented as a weighted graph where the vertices are the computational elements, and the weights on the edges indicate the number of registers between computational elements. A delay associated with each computational element is included as a vertex weight. The minimum clock period is the maximum sum of vertex weights between any pair of registers. The usual *ad hoc* pipelining techniques insert additional registers between computational elements to decrease the clock period. The retiming techniques try to find an optimal placement of the registers by phrasing this problem as a particular integer programming problem that can be solved quickly.

The general ideas behind retiming provide a broad framework in which clocked circuit optimization problems besides clock period minimization can be considered, and it allows one to bring the powerful combinatorial optimization techniques to bear on these problems. Retiming seems to be a valuable technique that could be incorporated into both circuit and compilers and interactive design tools.

3.8 Wafer-Scale Wiring

VLSI technologists are fast developing wafer-scale integration. Rather than partitioning a silicon wafer into chips as is usually done, the idea behind wafer-scale integration is to assemble an entire system (or network of chips) on a wafer, thus avoiding the costs and performance loss associated with individual packaging of chips. A major problem with assembling a large system of microprocessors on a single wafer, however, is that some of the processors are likely to be defective. Thus a practical procedure for integrating wafer-scale systems must have the ability to configure networks around faults.

Recently, a group headed by Tom Leighton has developed improved and provably efficient algorithms for constructing two-dimensional systolic arrays on a wafer [7, 8]. Systolic arrays are a desirable architecture for VLSI because all communication is between the nearest neighbors. In a wafer-scale system, however, all the nearest neighbors of a processor may be dead, and thus the prime advantage of adopting a systolic array architecture may be lost if a long wire connects electrically adjacent processors. In general, the longest interconnection between processors will be a bottleneck in the system. Of the many possible ways in which the live cells on a wafer can be connected to form a systolic array, therefore, the one that minimizes the length of the longest wire is most desirable.

The new algorithm for integrating two-dimensional arrays is based on a matching process. In particular,

we find a matching between the functioning processors and points in an imaginary grid with evenly spaced rows and columns for which the length of the longest distance between matched points is minimized. Assuming independent cell failures, we prove that the longest matching length is $O(\log^{3/4} N)$ with very high probability. Initial experimental evidence confirms that the matching distances are quite small, and that the wire lengths in the resulting $N \times N$ array are within a factor of two of $\log^{3/4} N$.

The new algorithm is dramatically superior to row/column elimination approaches to wafer-scale integration (which fail for large N and/or high probability of cell failure), and moderately better than previously discovered divide and conquer algorithms. The algorithm requires roughly $O(N^2)$ time for N processors in the worst case, but appears to run much faster on average.

3.9 Compaction

An automatic compaction procedure is an effective tool for cutting production costs of a VLSI circuit at low cost to the designer, because the yield of fabricated chips is strongly dependent on the total circuit area. In order to perform any sort of compaction, the components of the layout must be differentiated into *modules*, which are fixed in size and shape, and *wires*, which are flexible. Common procedures for generating design rule constraints [5, 6, 13] assume that wires are simply rectangular regions of variable length, and otherwise identical to modules. A vertical wire, for example, would be assigned an x-coordinate during horizontal compaction, and could only be moved rigidly from side to side. But one would often like a previously straight wire bent around an obstacle during compaction, if the are of the circuit could thereby be reduced.

This problem is not easily overcome. Many systems [5, 26] attempt to solve it by allowing the designer to specify *jog* point at which wires my bend. Compaction then becomes an interactive procedure in which the designer repeated examines the compacted layout, adds more potential jog points and retries the compaction. Miller Maley, a student of Charles Leiserson has developed a new polynomial-time compaction algorithm that automatic introduces jog points in an optimal fashion [14]. Automatic jog insertion is achieved by treating wires not as solid objects, but only as indicators of the topology of the layout. Constraints between modules no longer express design rules directly; instead, they will ensure that there exist paths for the wires, having the given topology that satisfy the rules. The new constraints, called *routability conditions*, can be formulated as simple linear inequalities, and are easily solved. When the optimal placements have been established, the new wire paths are determined by a single-layer router [9]. This router does not generate "unnecessary U's," and therefore minimises wire lengths, given that the layout topology is fixed.

Earlier work by Ron Rivest and his students introduced a theoretical approach to placement and routing. This work was embodied in a demonstration system called PI [20, 21], and is now being transferred to industry for practical implementation.

4. Conclusions

As with the larger MIT VLSI effort, we have stressed interdisciplinary approaches to VLSI CAD. This can be seen from the work described here: SCHEMA, which applies novel software engineering techniques to building a CAD frame; retiming, where integer programming ideas are applied to pipelined architectures; compaction, where we are using new combinatorial optimization techniques for jog insertion. The work on macromodels and reliabilty analysis is an example of tools that span the usual design boundaries. We are guided by the principle that the most challenging and needed work is that which crosses or even obliterates conventional design boundries.

We have summarized our most important recents results in VLSI CAD. Additional information on specific efforts may be found in the references.

5. Acknowledgments

This work was supported in part by the Air Force Office of Scientific Research, under contracts F29620–81–C–0054, OSR–82–0326 and F49620–84–C–0004; in part by DARPA under contract N00014–80–C–0622; in part by NSF grant ECS83–10941; in part by fellowships from the RCA Corporation, the Office of Naval Research, National Science Foundation, IBM Corporation, Bantrell Foundation and Analog Devices; and in part by the Harris Corporation.

6. References

1. C. S. Bamji, *A Design by Example Regular Structure Generator*, M.S. thesis, Dept. of Electrical En-

gineering and Computer Science, Massachusetts Institute of Technology, February, 1985.

2. G.C. Clark, R. Zippel, "Schema: An Architecture for Knowledge Based CAD," *Proceedings of IC-CAD'85*, (1985), 50–52.
3. T. S. Hohol, *RELIC: A Reliability Simulator for Integrated Circuits*, M.S. thesis, Dept. of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, (1986).
4. T. S. Hohol and L. A. Glasser, "RELIC: A Reliability Simulator for Integrated Circuits," *IEEE International Conference on Computer-Aided Design*, Santa Clara, CA, November, 1986.
5. M. Y. Hsueh, *Symbolic Layout and Compaction Of Integrated Circuits*, Ph. D. thesis, Dept. of Electrical Engineering and Computer Science, University of California, Berkeley, CA, (1979).
6. G. Kedem and H. Watanabe, *Optimisation Techniques for IC Layout and Compaction*, Technical Report 117, Computer Science Department, University of Rochester, September, 1982 .
7. F. T. Leighton and C. E. Leiserson, "Wafer-Scale Integration of Systolic Arrays," *IEEE Trans. on Computers*, C-34(5), (1984), 448–461.
8. F. T. Leighton and P. Shor, "Tight Bounds on the Complexity of Minimax Grid Matching with Applications to the Average Case Analysis of Algorithms," *Proc. ACM Symposium on the Theory of Computing*, (1986).
9. C. E. Leiserson and F. M. Maley, "Algorithms for Routing and Testing Routability of Planar VLSI Layouts," *17th Annual ACM Symposium on Theory of Computing*, May, 1985.
10. C. E. Leiserson, F. M. Rose and J. B. Saxe, "Optimizing Synchronous Circuitry by Retiming," *Third Caltech Conference on Very Large Scale Integration*, date of publication unknown, 87–116.
11. C. E. Leiserson and J. B. Saxe, "Optimizing Synchronous Systems," *Journal of VLSI and Computer Systems*, 1(1), Spring, 1983, 41–67.
12. C. E. Leiserson and J. B. Saxe, "Retiming Synchronous Systems," date of publication unknown.
13. T. Lengauer, "Efficient Algorithms for Constraint Generation for Integrated Circuit Layout Compaction," *Proceedings of the 9th Workshop on Graph Theoretic Concepts in Computer Science*, January, 1984.
14. F. M. Maley, "Compaction with Automatic Jog Introduction," *1985 Chapel Hill Conference on Very Large Scale Integration*, Computer Science Press, Rockville, MD, (1985), 261–283.
15. M. D. Matson and L. A. Glasser, "Macromodeling and Optimization of Digital MOS VLSI Circuit," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems.*, in preparation.
16. S. P. McCormick, *Automated Circuit Extraction from Mask Descriptions of MOS Networks*, M.S. thesis, Dept. of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, February, 1984.
17. S. P. McCormick, "EXCL, A Circuit Extractor for IC Designs," *Proceedings of 21st Design Automation Conference*, Albuquerque, NM, June, 1984, 616–623.
18. D. A. Moon, R. M. Stallman, D. L. Weinreb, *LISP Machine Manual, Fifth Edition*, MIT Artificial Intelligence Lab., Cambridge, MA, January, 1983.
19. P. O'Brien and J.L. Wyatt, Jr., "Signal Delay in ECL Interconnect," *Proc. 1986 IEEE Int. Symp. on Circuits and Systems*, San Jose, CA, May, 1986, 755–758.
20. R. L. Rivest, "The PI (Placement and Interconnect) System," *Proc. 19th Design Automation Conference*, Las Vegas, June, 1982, 475–481.
21. R. L. Rivest and C. M. Fiduccia, "A Greedy Channel Router," *Proc. 19th Design Automation Conference*, Las Vegas, June, 1982, 418–424.
22. J. Rubinstein, P. Penfield, Jr., and M.A. Horowitz, "Signal Delay in RC Tree Networks," *IEEE Trans. Computer-Aided Design*, CAD-2(3), July, 1983, 202–211.
23. D. Standley and J.L. Wyatt, Jr., *Improved Signal Delay Bounds for RC Tree Networks*, VLSI Memo No. 86–317, Microsystems Research Center, Room 39-321, MIT, Cambridge, MA 02139, May, 1986 .
24. S. Teig, R. Smith, and J. Seaton, "Timing Driven Layout of Cell-Based IC's," *VLSI Systems Design*, May, 1986, 63–73.
25. J. P. Wade and C. G. Sodini, "Dynamic Cross-Coupled Bitline Content Addressable Memory Cell for High Density Arrays," *Proceedings of IEDM*, (1985).
26. J. D. Williams, "STICKS—A Graphical Compiler for High Level LSI Design," *National Computer Conference*, (1978), 289–295.
27. J. L. Wyatt, Jr., "Signal Delay in RC Mesh Networks," *IEEE Trans. Circuits and Systems*, CAS-32(5), May, 1985, 507–510.
28. J.L. Wyatt, Jr., C. Zukowski, and P. Penfield, Jr., "Step Response Bounds for Systems Described by *M*-Matrices, with Application to Timing Analysis of Digital MOS Circuits," *Proc. 24th IEEE Conf. on Decision and Control*, Ft. Lauderdale, FL, December, 1985, 1552–1557.
29. Q. Yu, J.L. Wyatt, Jr., C. Zukowski, H.N. Tan, and P. O'Brien, "Improved Bounds on Signal Delay in Linear RC Models for MOS Interconnect," *Proc. IEEE 1985 Int. Symp. on Circuits and Systems*, Kyoto, Japan, June, 1985, 903–906.
30. R. Zippel, *The Database Accelerator: Architecture*, Smart Memories Project, Report 1, (1985) .
31. C.A. Zukowski, *The Bounding Approach to VLSI Circuit Simulation*, Kluwer Academic Publishers, July, 1986.

Parallel $(\Delta+1)$ Coloring of Constant-Degree Graphs

Andrew V. Goldberg*
Serge A. Plotkin†
Laboratory for Computer Science
MIT
Cambridge MA 02139

September 24, 1986

Abstract

This paper presents parallel algorithms for coloring a constant-degree graph with a maximum degree of Δ in $(\Delta+1)$ colors and for finding a maximal independent set in a constant-degree graph. Given a graph with n vertices, the algorithms run in $O(\lg^* n)$ time on EREW PRAM with $O(n)$ processors. The algorithms use only local communication and achieve the same complexity bounds when implemented in the distributed model of parallel computation.

Keywords: parallel algorithms, graph algorithms, graph coloring, maximal independent set.

1 Introduction

Both the maximal independent set (MIS) problem and the problem of vertex-coloring of a graph with a maximum degree of Δ in $(\Delta+1)$ colors (denoted by VC throughout this paper) are known to be in the class NC of problems that can be solved in polylogarithmic time using a polynomial number of processors. Polylogarithmic MIS algorithms are described in [7,8] and polylogarithmic graph coloring algorithms are described in [6]. The best previous parallel MIS algorithm is due to Luby [8]. The deterministic version of this algorithm runs in $O(\lg^2 n)$ EREW PRAM time using $O(n^2 m)$ processors. (Throughout this paper we use n to denote the number of vertices, and m to denote the number of edges in the input graph.) The total number of operations in this algorithm

*Supported by a Fannie and John Hertz Foundation Fellowship and by the Defense Advanced Research Projects Agency under the contract N00014-80-C-0622

†This work was supported in part by the Advanced Research Projects Agency under the Office of Naval Research contract N00014-75-C-0661

(given by the processor-time product) is $O(n^2 m \lg^2 n)$, which is a large number compared to $O(m)$ operations of the sequential algorithm. The graph coloring algorithms described by Karloff in [6] make use of Luby's MIS algorithm as a subroutine and therefore have a large processor-time product as well.

In this paper we present algorithms for the MIS and VC problems that work well for constant-degree graphs. For these graphs, the algorithms run in $O(\lg^* n)$ time using $O(n)$ processors, and the number of sequential operations of the algorithms is almost linear, namely $O(n \lg^* n)$. Our approach is a generalization of the deterministic coin-flipping technique of Cole and Vishkin [3]. Although their technique, like ours, can be viewed as an iterative improvement of coloring, it works only for directed chains whereas our method works for any constant-degree graph.

The algorithms presented in this paper can be implemented in the distributed model of computation [4,2], where processors have fixed connections determined by the input graph. The algorithms in the distributed model achieve the same $O(\lg^* n)$ bound as in the EREW PRAM model. Awerbuch has recently shown [1] using Ramsey theory that $\Omega(\lg^* n)$ time is required in the distributed model to find a maximal independent set on a chain. Our algorithms are therefore optimal (to within a constant factor) in the distributed model.

2 Definitions and Notation

This section defines the MIS and VC problems, describes the assumptions about the computational model, and introduces the notation used throughout the paper.

In this paper we use n to denote the number of vertices and m to denote the number of edges in the graph. We use Δ to denote the maximum degree of the graph.

The two problems discussed in this paper are the maximal independent set (MIS) and vertex-coloring (VC) problems. The MIS problem is, given an undirected graph $G = (V, E)$, to find a maximal set of vertices $I \subseteq V$ such that no two vertices in I are adjacent. A valid coloring of a graph is an assignment of a color to each vertex such that no two neighboring vertices have the same color. The vertex-coloring (VC) problem is to find a valid coloring that uses at most $\Delta + 1$ colors.

We use the following standard notation:

$$\begin{aligned} \lg x &= \log_2 x \\ \lg^{(1)} x &= \lg x \\ \lg^{(i)} x &= \lg \lg^{(i-1)} x \\ \lg^* x &= \min\{i \mid \lg^{(i)} x \leq 2\} \end{aligned}$$

We assume an exclusive-read, exclusive-write (EREW) PRAM model of computation where each processor is capable of executing simple word and bit operations. The word width is assumed to be $\lceil \lg n \rceil$. The word operations we use include bit-wise boolean operations, integer comparisons, and unary-to-binary conversion. In addition, each processor has a unique identification number (PE-ID).

3 The Algorithms

This section presents three algorithms. Given a graph with a constant degree Δ, the first algorithm colors it into a constant number of colors. The second algorithm uses this coloring and finds a maximal independent set in the graph. The third algorithm iteratively applies the second one to find an MIS in the graph and recolors the graph using only $\Delta+1$ colors. All these algorithms run in $O(\lg^* n)$ time on an EREW PRAM.

We first describe an $O(\lg^* n)$ time algorithm for coloring a constant-degree graph using a constant number of colors. The algorithm accepts as input a graph $G = (V, E)$ whose maximum degree is a constant Δ. Let C be an initial coloring of the graph. One possible way to obtain an initial coloring is to assign each vertex v a color C_v that is equal to the unique identification number of the processing element associated with v (denoted by PE-ID(v) in figure 1).

The main idea of the algorithm is to iteratively improve the coloring by constructing, for every vertex, a list of differences between the vertex and its neighbors and using this list as a new color for the vertex. Each element of the list is an ordered pair that consists of the index of a position in which the color of the vertex differs from the colors of one of its neighbors and the value of the bit at this position. The index can be represented by a string of bits of length $\lceil \lg L \rceil$ where L is the length of the representation of the current colors. Since the maximum degree is a constant, the length of the new representation is $O(\lg L)$.

The algorithm is shown in figure 1 and works as follows. Starting from a valid coloring C, we iteratively recolor the graph, each time reducing the number of colors. For each vertex v and each

```
PROCEDURE Color-Constant-Degree-Graph
L ← ⌈lg n⌉
for all v ∈ V in parallel set C_v ← PE-ID(v)   ;;; initial coloring
while L > Δ⌈lg L + 1⌉ for all v ∈ V in parallel do
      N_v ← {w | (v, w) ∈ E}
      for all w_k ← N_v(k), 1 ≤ k ≤ |N_v| do   ;;; find the index of diff. bit
          i_k ← min{i | C_v(i) ≠ C_{w_k}(i)}
          b_k ← C_v(i_k)
      end
      for all |N_v| < k ≤ Δ do
          i_k ← 0
          b_k ← C_v(0)
      end
      C_v ← i_1 b_1 i_2 b_2 ... i_Δ b_Δ   ;;; calculate new color
      L ← Δ⌈lg L + 1⌉
end
```

Figure 1: The Coloring Algorithm for Constant-degree Graphs

neighbor w of v we find an index i_w of the first bit in which C_v and C_w differ and construct a pair $\langle i_w, C_v(i_w)\rangle$. Such an index always exists because of the validity of the initial coloring. The new color of v is constructed by concatenating these pairs computed for all the neighbors of v.

Theorem 1 *The algorithm* Color-Constant-Degree-Graph *produces a valid coloring of a constant-degree graph in a constant number of colors in* $O(\lg^* n)$ *time on EREW PRAM.*

Proof: First we prove by induction that the coloring computed by the algorithm is valid, and afterwards we prove the upper bound on the execution time.

Each processor has a unique identification number and therefore the initial coloring is valid. We must show that if the coloring at the beginning of an iteration is valid, then the coloring calculated at the end of the iteration is also valid. Since the initial coloring is valid, for each pair of adjacent vertices v and w there exists at least one index i_w such that the colors C_v and C_w differ in bit position i_w. For each vertex v, the algorithm constructs a new color that can be viewed as a list of pairs of the form $\langle i_w, C_v(i_w)\rangle$, one pair per neighbor. In order for these lists to constitute a valid coloring, any two neighboring vertices v and w must have at least one different pair. Assume that the list constructed by the vertex v is

$$(\langle i_1, C_v(i_1)\rangle\langle i_2, C_v(i_2)\rangle \ldots \langle i_\Delta, C_v(i_\Delta)\rangle)$$

and the list constructed by the vertex w is

$$(\langle j_1, C_w(j_1)\rangle\langle j_2, C_w(j_2)\rangle \ldots \langle j_\Delta, C_w(j_\Delta)\rangle).$$

If for some k, $i_k \neq j_k$, we are done. On the other hand, if for all $1 \leq k \leq \Delta$, $i_k = j_k$, by the algorithm there exists a pair $\langle i_k, C_v(i_k)\rangle$ where i_k is an index of some bit position in which the colors C_v and C_w differ. Therefore, in this case, $C_v(i_k) \neq C_w(j_k)$. Hence, the constructed lists are always different for any two neighbors.

Each iteration of the algorithm is dominated by the computation of the index in which the colors of two neighboring vertices differ. Vishkin [3] shows how to find the minimum such index in constant time using word operations described in the previous section.

Now we show that the algorithm terminates in at most $O(\lg^* n)$ iterations. Let L_k denote the number of bits in the representation of colors after k iterations. For $k = 1$ we have

$$\begin{aligned} L_1 &= \Delta\lceil \lg L + 1\rceil \\ &\leq 2\Delta\lceil \lg L\rceil \end{aligned}$$

if $\lceil \lg L\rceil \geq \lceil \lg \Delta\rceil + 2$.
Assume that for some $k-1$ we have

$$\begin{aligned} L_{k-1} &\leq 2\Delta\lceil \lg^{(k-1)} L\rceil \\ \lceil \lg^{(k)} L\rceil &\geq \lceil \lg \Delta\rceil + 2. \end{aligned}$$

Then

$$\begin{aligned} L_k &= \Delta\lceil \lg L_{k-1} + 1\rceil \\ &\leq \Delta(\lceil \lg(2\Delta \lg^{(k-1)} L)\rceil + 1) \\ &\leq 2\Delta\lceil \lg^{(k)} L\rceil \end{aligned}$$

Therefore, as long as $\lceil \lg^{(k)} L\rceil \geq \lceil \lg \Delta\rceil + 2$,

$$L_k \leq 2\Delta\lceil \lg^{(k)} L\rceil.$$

Hence, the number of bits (L_k) in the representation of colors decreases until, after $O(\lg^* n)$ iterations, it reaches the value of $2\Delta\lceil \lg \Delta + 2\rceil$ or less, which is constant. The algorithm terminates at this point. ∎

Given a coloring of a graph in a constant number of colors, we can directly find an MIS in the graph.

Theorem 2 *In a constant-degree graph, MIS can be found in $O(\lg^* n)$ time on EREW PRAM using $O(n)$ processors.*

Proof: First, we use the algorithm *Color-Constant-Degree-Graph* to color the input graph in a constant number of colors. After the graph is colored, we iterate over all the colors, selecting the remaining vertices of the current color, adding them to the independent set and deleting all their neighbors from the graph. It can be shown by induction on the number of colors that this procedure produces an MIS.

The running time of this MIS algorithm is dominated by the *Color-Constant-Degree-Graph* subroutine: each iteration takes constant time, and the number of iterations is equal to the total number of colors and therefore is a constant. ∎

The above MIS algorithm can be used to produce a $(\Delta+1)$ vertex-coloring of the graph.

Theorem 3 *A graph whose maximum degree is a constant Δ can be colored in $\Delta+1$ colors in $O(\lg^* n)$ time on EREW PRAM using $O(n)$ processors.*

Proof: The Δ+1 coloring algorithm works by iteratively finding an MIS in the input graph, coloring the MIS with a new color, and deleting it from the graph. If a vertex v is not removed at the end of some iteration, at least one of its neighbors is removed during this iteration. Therefore, after each iteration, the maximum degree of the graph induced by the remaining vertices decreases. Hence, after at most $\Delta+1$ iterations, the algorithm terminates producing a $\Delta+1$ coloring of the graph.

Each iteration of the algorithm takes $O(\lg^* n)$ time (by theorem 2), and the number of iterations is constant. Therefore, the overall running time of the algorithm is $O(\lg^* n)$. ∎

4 Discussion and Further Results

In this paper we have described a new technique for deterministic symmetry-breaking and have shown how to apply this technique to vertex-coloring and maximal independent set problems.

The algorithms presented in this paper have good asymptotic behavior, but the constant factors are large because the procedure *Color-Constant-Degree-Graph* uses a large number of colors compared to the graph degree. In [5] we describe a somewhat more complicated algorithm that uses significantly less colors.

The techniques used in this paper can be further extended. In combination with other methods, these techniques can be used to obtain polylogarithmic time linear processor algorithms for finding an MIS and a $(\Delta+1)$ vertex-coloring in a wide class of graphs that includes trees, planar graphs, graphs with polylogarithmic thickness (i.e. graphs that are unions of polylogarithmic number of planar graphs), and graphs with polylogarithmic maximum degree; see technical memorandum [5] for details. The memorandum also gives an $O(\lg^* n)$ algorithm for 3-coloring of a rooted tree, an $O(\lg n \lg^* n)$ algorithm for 7-coloring of a planar graph, and proves an $\Omega(\lg n / \lg\lg n)$ lower bound on any algorithm in CRCW PRAM model that 2-colors a rooted tree using a polynomial number of processors. The same lower bound is proven for the general MIS problem.

5 Acknowledgment

We would like to thank Charles Leiserson for fruitful and stimulating discussions, and for his valuable comments on a draft of this paper.

References

[1] B. Awerbuch. 1986. Personal Communication.

[2] B. Awerbuch. Complexity of network syncronization. *Journal of the Association for Computing Machinery*, 32(4):804–823, October 1985.

[3] R. Cole and U. Vishkin. Deterministic coin tossing and accelerating cascades: micro and macro techniques for designing parallel algorithms. In *Proc. 18th ACM Simp. on Theory of Computing*, pages 206–219, 1986.

[4] R. G. Gallager, P. A. Humblet, and P. M Spira. A distributed algorithm for minimum-weight spanning trees. *ACM Transactions on Programming Languages and Systems*, 5(1):66–77, January 1983.

[5] A. Goldberg and S. Plotkin. *Efficient Parallel Coloring and Maximal Independent Set Algorithms.* Technical Report, MIT, 1986. (To appear).

[6] H. J. Karloff. *Fast Parallel Algorithms for Graph-Theoretic Problems: Matching, Coloring, Partitioning.* PhD thesis, University of California, Berkeley, 1985.

[7] R. M. Karp and A. Wigderson. A fast parallel algorithm for the maximal independent set problem. In *Proc. 16th ACM Simp. on Theory of Computing*, pages 266–272, 1984.

[8] M. Luby. A simple parallel algorithm for the maximal independent set problem. In *Proc. 17th ACM Simp. on Theory of Computing*, pages 1–10, 1985.

END

5-87

DTIC