Circuit placement, chip optimization, and wire routing for IBM IC technology

by D. J. Hathaway
R. R. Habra
E. C. Schanzenbach
S. J. Rothman

Recent advances in integrated circuit technology have imposed new requirements on the chip physical design process. At the same time that performance requirements are increasing, the effects of wiring on delay are becoming more significant. Larger chips are also increasing the chip wiring demand, and the ability to efficiently process these large chips in reasonable time and space requires new capabilities from the physical design tools. Circuit placement is done using algorithms which have been used within IBM for many years, with enhancements as required to support additional technologies and larger data volumes. To meet timing requirements, placement may be run iteratively using successively refined timing-derived constraints. Chip optimization tools are used to physically optimize the clock trees and scan connections, both to improve clock skew and to improve wirability. These tools interchange sinks of equivalent nets, move and create parallel copies of clock buffers, add load circuits to balance clock net loads, and generate balanced clock tree routes. Routing is done using a grid-based, technology-independent router that has been used over the years to wire chips. There are numerous user controls for specifying router behavior in particular areas and on particular interconnection levels, as well as adjacency restrictions.

Introduction

Traditionally, the goals of chip physical design have been to find placements which are legal (i.e., are in valid locations and do not overlap each other) and wirable for all circuits in a fixed netlist, and to route wires of uniform width on a small number of layers (two or three) to complete the interconnections specified in that netlist. The physical design process has been divided into two parts:

©**Copyright** 1996 by International Business Machines Corporation. Copying in printed form for private use is permitted without payment of royalty provided that (1) each reproduction is done without alteration and (2) the *Journal* reference and IBM copyright notice are included on the first page. The title and abstract, but no other portions, of this paper may be copied or distributed royalty free without further permission by computer-based and other information-service systems. Permission to *republish* any other portion of this paper must be obtained from the Editor.

0018-8646/96/$5.00 © 1996 IBM

placement, which is the assignment of circuits in the netlist to locations, or cells, on the chip image, and wiring, which is the generation of routes, using the available interconnection layers, to complete the connections specified in the netlist.

Recently, new technology characteristics and constraints and increased performance pressures on designs have required new capabilities from the chip physical design process. Wiring is now the dominant contributor to total net load and delay, and its contribution may vary significantly depending on the physical design solution chosen. This requires timing controls [1–4] for placement and wiring. Newer and larger chip technologies also provide more layers of wiring which must be accommodated by the wiring programs. These large chips also typically contain tens of thousands of latches, each requiring scan and clock connections. Such connections, as they appear in the input netlist to physical design, are usually somewhat arbitrary. Reordering the scan chain and rebuilding the clock distribution tree to reduce wire demand can significantly improve the physical design, since even with increased wiring layers these chips tend to be wire-limited. Clock trees must also be optimized to minimize clock skew, which has a direct impact on chip performance. Physical constraints on wire length and width to avoid electromigration failures and to limit noise must also be taken into consideration.

Hierarchical design of these large chips also imposes some new requirements on the physical design of the hierarchical components. However, in this paper we generally concentrate on the physical design of a single hierarchical component; other consequences of hierarchy are addressed in [1].

The design tools and the methodology for their use described in this paper have evolved from those used for earlier IBM technologies [2–4].

Physical design methodology

Many interdependencies exist among placement, clock and scan optimization, wiring, and hierarchical design planning [1]. Ordering of steps in the physical design process is required in order to give the best results and to ensure that the necessary prerequisites for each step are available. The general flow is as follows:

1. Identify connections to be optimized after placement, so that they will not influence placement. These include the scan and clock connections to latches.
2. Generate constraints for placement on the basis of a timing analysis done using idealized clock arrival times at latches and estimates of wire load and *RC* delay before physical design. These constraints include limits on the capacitance of selected nets and limits on the resistance or *RC* delay for selected connections.
3. Perform an initial placement to determine an improved basis for constraint generation, and optionally to fix the placement of large objects.
4. Generate new constraints for placement on the basis of a timing analysis done using wire load and *RC* delay values derived from the initial placement.
5. Perform placement.
6. Optimize the clock trees and scan connections.
7. Make logic changes, including changes to circuit power levels, to fix timing problems.
8. Legalize placement.
9. Generate new timing constraints for wiring on the basis of a timing analysis done using the actual clock tree and wire load and *RC* delay values derived from the final placement.
10. Perform routing.

Note that evaluation of the timing is performed at many points in this process, and the results determine whether to proceed to the next step or to go back through some of the previous steps. In particular, the user may need to iterate on constraint generation, placement, optimization, and timing until the design meets its timing goals. The user must also evaluate the wirability of the design throughout the process, and make adjustments to constraints or methodology if necessary.

Placement

Placement can be used at several points in the design process, and different algorithms are appropriate depending upon the state of the design.

Placement is often run before the logic has been finalized to obtain an early indication of the timing and wirability. At this point, the feedback may be used to influence logic changes. This may also be the time at which the locations of large objects are determined. The placement program may run more quickly by not considering such details as legality, and there may be less emphasis on achieving the best possible result. The results of this placement may be used as input to the tool which generates capacitance constraints used to drive subsequent placements.

Legality incorporates such constraints as the circuits not overlapping one another and remaining within the bounds of their placement area, being placed in valid orientations and in rows specified in the chip image, satisfying other restrictions supplied by either the user or the technology supplier, and ensuring that there are no circuit-to-power shorts (a concern in some custom circuits).

In the past, all legal location restrictions were specified to the placement programs in the form of "rules" which specify for a particular chip image and circuit type where on the chip circuits of this type can be placed. Now the program is expected, in most cases, to determine this

itself, in part because of the extensive number of available chip images and the large amount of data which might be involved.

Once the logic has stabilized, more emphasis is placed on achieving a high-quality, and legal, placement. Some placement tools ignore at least some aspects of legality during the optimization phase, relying upon a separate legalization postprocessing step. Others attempt to ensure that they produce a completely legal result, while permitting such conditions as overlaps (with penalty) during the optimization.

Both clock optimization and power optimization (switching implementations of circuits in order to improve timing) can produce overlaps. These overlaps can simply be removed through a "brute force" technique, or overlap removal can be performed with some form of placement optimization. It is important to ensure that the quality of the placement is maintained: Clock skew, timing, and wirability should not worsen. It is often necessary to compromise between these conflicting factors. For example, the smallest clock skew is achieved by preventing the circuits in timing-critical clock trees from moving during overlap removal, but this can cause the other circuits to move much farther and can affect both the timing and the wirability.

The basic algorithms used in our placement programs are simulated annealing [5] and quadratic placement with iterative improvement [6, 7]. These are by no means new techniques, but the programs have been continually enhanced to give better results, in general, and to support the new specific technology-driven requirements. For example, the simulated annealing placement program now has the capability of performing low-temperature simulated annealing (LTSA). LTSA determines the temperature at which an existing placement is in equilibrium, and starts cooling from that temperature, thus effecting local improvements to a placement without disrupting the global placement characteristics.

Both simulated annealing and quadratic placement accept many controls. They include preplacement, floor-planning, specification of circuits to be placed in adjacent locations, net capacitance and source-to-sink resistance constraints, and weights for the various components of the scoring function (including net length, congestion, and population balancing).

Chip optimization

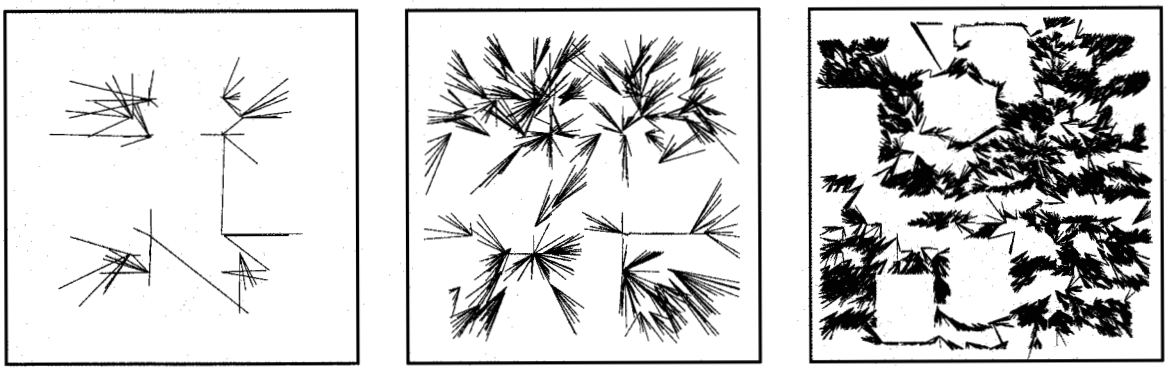
Generally, the netlist which is the input to the physical design process contains all connections and circuits required in the design, and must be preserved exactly through the physical design process. Connections within clock trees (and other large signal-repowering trees) and latch scan chains (and other types of serial connections such as driver inhibit lines), however, may be reconfigured to improve chip wirability and performance. The best configuration of these connections depends on the results of chip placement, and thus the final construction of these types of structures must be a part of the physical design process. We call these special physical design processes *chip optimization*.

Chip optimization consists of two major parts. First, because many of the connections in the portions of the design being optimized will change after placement, they must be identified before placement is done and communicated to the placement tools so that they do not influence the placement process. We call this process *tracing*. Second, after placement is done we must actually perform the optimization of these special sections of logic. The specific optimization steps differ for clock trees and for scan chains.

Tracing and optimization of clock trees have been done for several years using separate programs. Recently these functions have been taken over by a new combined clock tracing and optimization program. The tracing function in the earlier tool is essentially the same as that in the new one. The optimization capability, however, has been significantly enhanced. The earlier clock optimization program could interchange connections of equivalent nets (as identified by the tracer) using a simulated annealing algorithm, could move dummy load circuits (terminators), and could move driving buffer circuits to the center of the sinks being driven. All of these actions were performed to reduce wiring and to balance the load and estimated *RC* delay on equivalent nets. In the remainder of this paper we describe the capabilities of and results from the new combined tracing and optimization program when discussing clock tree optimization.

Tracing of clock trees takes as its input a list of starting nets (the roots of the clock tree) and a description of the stopping points. Tracing proceeds forward through all points reachable in a forward trace from the starting nets and stops when latches or other explicitly specified types of circuits are reached, or when other explicitly specified stopping nets are reached. Placement is told to ignore all connections within the clock tree.

Tracing of scan chains takes as its input a list of connections to be kept and a list of points at which the chains should be broken. Tracing proceeds by finding the scan inputs of latches and tracing back from them, through buffers and inverters if present, to their source latches. These scan connections are then collected into chains. Placement is told to ignore all connections in the scan chains which will be subject to reordering, and the list of these scan chain connections and the polarity of each (the net inversion from the beginning of the scan chain) are passed as input to the scan optimization program.



(a) (b) (c)

Figure 1

Load-balanced clock nets for levels (a) 1, (b) 2, and (c) 3.

A variety of styles of clock distribution network have been described in recent years. Several of these styles use a single large driver or a collection of drivers to drive a single clock net. Mesh clock distribution [8] and trunk and branch distribution [8] methods attempt to minimize clock skew by directly minimizing delay. This requires wide clock wiring (and/or many clock wires in the case of mesh distribution), thus causing a significant impact on wirability, and significant power expenditure to switch the high-capacitance clock net. H-tree [9] and balanced wire tree distribution [10–12] methods attempt to equalize the *RC* delay to all clock sinks using a delay-balanced binary tree distribution network. These methods tend to create long clock distribution delays owing to long electrical paths to the clock sinks. To avoid current density limitations of the clock conductors and excessive clock pulse degradation, these methods generally also require wide nets toward the root of the clock tree, again affecting wirability and power consumption. The delay problems of the single net distribution schemes are basically due to the $O(n^2)$ increase of *RC* delay with wire length. By limiting the length and load of any individual clock net in the clock distribution tree, this behavior is eliminated. For these reasons, our clock optimization methodology is directed toward a distributed buffer tree clock distribution network [10, 13].

The goals of the optimization vary for different levels of the clock tree. Toward the root, where the interconnection distances are large (and hence the *RC* delay is significant) and the number of nets is small, *RC*-balanced binary tree routing is used to help balance skew. Toward the leaves, where interconnection distances are very small (and hence *RC* delays are negligible) and where the number of nets is large, normal minimum Steiner routing is used, and the optimization goal is to balance the net loadings in order to balance the driving circuit delays. Because balanced tree routing requires more wiring resource than minimum Steiner routing, this approach tends to improve chip wirability.

Optimization of any fan-out tree always has as one goal the minimization of wiring congestion. For clock trees, an additional (and often more important) goal is the minimization of clock skew. The clock optimization performed includes the interchange of equivalent connections, the placement of circuits in the clock tree, the adjustment of the number of buffers needed in the clock tree, and the generation of balanced wiring routes for skew control. The new clock tracing and optimization program is designed as a collection of optimization algorithms which are called out by a Scheme language [14] script which is modifiable by the user. New features include the following:

- It can directly optimize a cross-hierarchical clock tree.
- It can add and delete terminators to better balance the capacitive load.
- It can make parallel copies of clock buffers. This means that the netlist can start with a skeleton clock tree that has the correct number of levels, but only one buffer at each level, and the optimizer will fill out the tree with the necessary number of buffers at each stage.
- It has an option to generate balanced wire routes for long skew-critical nets. This option creates "floor-plan routes" which are subsequently embedded in detail by the wiring program. By avoiding the issues of detailed wiring in the optimizer, we eliminate the data volume

required for detailed blockage information, which in turn makes it easier to perform cross-hierarchy optimization.
- It operates in several passes from the leaves to the root of the clock tree, allowing it to consider the locations of both inputs (established during the previous pass) and outputs of a block when determining its location.
- A combination of greedy initialization and iterative improvement functions offers performance improvements over the simulated annealing algorithm used in the previous clock optimization tool.

An example of the results of load balancing is shown in **Figure 1**. The three parts of the figure illustrate the three levels of a clock tree on an IBM Penta technology [15] chip containing 72000 circuits and 13000 latches, and occupying 713000 image cells on a 14.5-mm image. The characteristics of the resultant trees, before addition of dummy loads for final load balancing, are shown in **Table 1**.

Scan chain optimization is performed using a simulated annealing algorithm to reconfigure the connections in each chain in order to minimize wire length. If the user has specified breaks in the chain, the program optimizes each section of the chain separately. The program also preserves the polarity of each latch in a scan chain. Each latch is connected such that the parity (evenness or oddness) of the number of inversion between it and the start of the chain is preserved. Future work in this area will replace the simulated annealing optimization algorithm with a greedy initialization function followed by an iterative refinement step, in a manner similar to that employed in the new clock optimization program.

Routing

The routing program [16] has evolved over the years in response to a variety of pressures. With improvements in devices, routing plays an increasingly larger part in the design performance. Users need tighter control over the routing to improve the design and achieve greater productivity. The routing program has also had to handle the rapid increases in chip sizes and density.

As circuits become faster and wires become narrower, wires comprise a much larger part of path delays. Before routing, timing analysis is run using estimated paths. On the basis of this analysis, capacitance limits are generated for the critical nets and used by the routing program. In resolving congested areas, the capacitance of these critical nets is not allowed to exceed the limits. Less critical nets are rerouted around the area of congestion.

The routing program receives guidance from the clock optimization program for nets in clock and other timing-critical trees, in the form of floorplan routes. The routing program breaks each of these multi-pin nets into a group of point-to-point subnets. Each of these subnets is then routed to match the delay selected by the clock optimization program as closely as possible.

Table 1 Clock tree load-balancing results.

Tree level	*Number of nets*	*Estimated net load (fF)*			
		Maximum	*Minimum*	*Mean*	*Standard deviation*
1	24	1142	731	947	112
2	123	1446	773	1078	108
3	1120	646	285	529	20

To achieve the desired electrical and noise characteristics, users can specify the wire width and spacing to be used for each net. Noise becomes a problem when the switching of one net causes a significant change in voltage on an adjacent net because of capacitive coupling. Clock nets are often given a wider width and spacing to reduce their resistance, capacitance, and noise.

High clock speeds and long narrow wires can result in a reliability problem known as electromigration. Over time, the movement of electrons can move the metal atoms and result in a break in the wire. To avoid this problem, the nets are evaluated prior to routing to determine which are susceptible to electromigration failure. These nets are then assigned capacitance limits and may be assigned a greater wire width.

Users often want to fine-tune the wires for some nets, such as clocks, and keep these wires fixed through multiple passes of engineering changes. Users would also like to stop between iterations of routing to verify that the routing of the selected nets has met all criteria before continuing. To accommodate these requirements, the routing program allows nets and wire segments to be assigned to groups. The user can specify how to treat existing wires on the basis of the group they are in. For each iteration, all existing wires in a group can be

- Fixed (not allowed to be rerouted).
- Fixed unless erroneous (segments which are invalid after an engineering change can be rerouted).
- Allowed to be rerouted if needed to complete another connection.
- Deleted (in the case of a major logic or placement change).

At the end of routing, all new wire segments are assigned to a user-specified group. The routing program makes sure that nets routed in one iteration do not prevent the remaining nets from being completed. This allows the user to have the program route just the clock nets in the first iteration. Once it has been verified that

these routes meet the clock skew objectives, the wires for these nets can be fixed during the remaining iterations. A set of timing-critical nets can be routed in the second iteration. After analysis has verified that these nets meet their timing objectives, the remaining nets can be routed in the third iteration without changing the wires for the clock and timing-critical nets. This methodology allows tight clock skew and timing objectives to be met; it also allows timing problems requiring logic or placement changes to be identified quickly, before running a relatively long routing iteration on the majority of the nets.

Current chips can measure over twenty millimeters on a side and contain up to six layers of routing requiring 1600 megabytes to describe if kept in an uncompressed format. Designs can contain over a third of a million nets and a million pins which must be connected with over 300 meters of wire. The routing program uses compressed forms of the image, pin, and wire data in order to reduce system requirements and be able to handle these large designs on a workstation, even in flat mode. The 1600-megabyte chip description can be compressed to three megabytes. The data representation of 300 meters of wire, made up of over three million wire segments and two million vias, can be compressed to only 35 megabytes.

Before starting a potentially long routing run on a large design, the routing program allows the user to evaluate the design. A fast global routing step can be run to identify areas of congestion which may have to be resolved by changing the placement. The global results can also be fed to timing analysis to determine whether placement or logic changes must be made before detailed routing should be started. A single iteration of detailed routing can also be run to help identify congestion and timing problems before making a full routing run. A special iteration of routing can be made to identify pins which are inaccessible because of errors in the design rules, placement, or power routes.

Logic and placement are often changed to improve the design after the first routing run. The routing program automatically determines how these changes affect the wires and makes the required updates. This includes detecting old wires which are now shorted to new or moved circuits. The checking and update phases of the routing program run quickly when the logic and placement changes have been limited to small areas.

The user can control the cost of routing in each direction by interconnection level for up to four groups of nets. This can be used to have the short nets prefer the lower interconnection levels and the long nets use the upper interconnection levels. These weights can be set by area. This method is useful between macros where there is a high demand parallel to the edges of the macros and little demand to enter the macros.

In addition to congestion, timing, clock skew, and data volume, the routing program must handle special features of the technology. The routing program is often given multiple points at which it can connect to a pin. These points are in groups connected through high-resistance polysilicon. The routing program is prevented from routing into one group of a pin and out another, so that there is no polysilicon in the middle of a path to adversely affect timing and reliability. Unused pins must be connected to power or ground. The routing program recognizes any unused pins and ties them to the proper power bus.

If the routing program cannot resolve all of the congestion and complete all connections, a "ghost" iteration is run. This iteration completes as much of each of the remaining connections as possible and routes special wires, flagged as "ghosts," where no room can be found. The ghost wires may be replaced manually or automatically using a new set of parameters. Timing analysis can be run using these ghost wires as estimates. Display of the ghost wires can help identify congested areas.

Summary

Changes in physical design tools and methodology have been made to accommodate the higher performance requirements, larger chip sizes, and increasing importance of interconnect delay found in today's chip designs. Enhancements have been made to the placement, chip optimization, and routing tools to improve their capacity and performance and the quality of their results. Controls and options have been added to the tools to help the designer iteratively converge on a viable physical design implementation. The tools have also been enhanced to accommodate new requirements imposed by the technology.

The placement, clock optimization, and routing tool described here have been used on numerous timing-critical CMOS designs. Clocks for these designs range from 50 MHz up to 250 MHz. The clock skew due to physical design has been under 200 ps, although the skew due to process, power supply, and other variation can be ten times that. As an example, a design with 206 000 objects to be placed and 205 000 nets to be routed has been completed using a 15.5-mm chip image; it used more than 130 meters of wire and 1.6 million vias. Without clock and scan optimization, this design might have used more than 200 meters of wire, requiring a larger chip image.

Acknowledgments

The authors wish to acknowledge the contributions of Roger Rutter of IBM Endicott, NY, for his contributions to the chip optimization methods described here, and Chuck Meiley of IBM Almaden, CA, for his contributions to the wiring methods described here and for his

assistance with the wiring portions of this paper. We also thank Bruce Winter of IBM Rochester, MN, for his assistance in providing design examples used in this paper, and both Bob Lembach of IBM Rochester, MN, and Mike Trick of IBM Burlington, VT, for their methodology descriptions.

References

1. J. Y. Sayah, R. Gupta, D. Sherlekar, P. S. Honsinger, S. W. Bollinger, H.-H. Chen, S. DasGupta, E. P. Hsieh, E. J. Hughes, A. D. Huber, Z. M. Kurzum, V. B. Rao, T. Tabtieng, V. Valijan, D. Y. Yang, and J. Apte, "Design Planning for High-Performance ASICs," *IBM J. Res. Develop.* **40,** No. 3, 431–452 (1996, this issue).
2. R. S. Belanger, D. P. Conrady, P. S. Honsinger, T. J. Lavery, S. J. Rothman, E. C. Schanzenbach, D. Sitaram, C. R. Selinger, R. E. DuBois, G. W. Mahoney, and G. F. Miceli, "Enhanced Chip/Package Design for the IBM ES/9000," *Proceedings of the IEEE International Conference on Computer Design*, 1991, pp. 544–549.
3. J. H. Panner, R. P. Abato, R. W. Bassett, K. M. Carrig, P. S. Gillis, D. J. Hathaway, and T. W. Sehr, "A Comprehensive CAD System for High-Performance 300K-Circuit ASIC Logic Chips," *IEEE J. Solid-State Circuits* **26,** No. 3, 300–309 (March 1991).
4. R. F. Lembach, J. F. Borkenhagen, J. R. Elliot, and R. A. Schmidt, "VLSI Design Automation for the Application System/400," *Proceedings of the IEEE International Conference on Computer Design*, 1991, pp. 444–447.
5. S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by Simulated Annealing," *Science* **220,** No. 4598, 671–680 (May 1983).
6. K. J. Antreich, F. M. Johannes, and F. H. Kirsch, "A New Approach for Solving the Placement Problem Using Force Models," *Proceedings of the IEEE Symposium on Circuits and Systems*, 1982, pp. 481–486.
7. R.-S. Tsay, E. S. Kuh, and C.-P. Hsu, "PROUD: A Fast Sea-of-Gates Placement Algorithm," *Proceedings of the 25th ACM/IEEE Design Automation Conference*, 1988, pp. 318–323.
8. K. Narayan, "Clock System Design for High Speed Integrated Circuits," *IEEE/ERA Wescon/92 Conference Record*, 1992, pp. 21–24.
9. H. B. Bakoglu, J. T. Walker, and J. D. Meindl, "A Symmetric Clock Distribution Tree and Optimized High Speed Interconnections for Reduced Clock Skew in ULSI and WSI Circuits," *Proceedings of the IEEE International Conference on Computer Design*, 1986, pp. 118–122.
10. K. M. Carrig, D. J. Hathaway, K. W. Lallier, J. H. Panner, and T. W. Sehr, "Method and Apparatus for Making a Skew-Controlled Signal Distribution Network," U.S. Patent 5,339,253, 1994.
11. R.-S. Tsay, "An Exact Zero-Skew Clock Routing Algorithm," *IEEE Trans. Computer-Aided Design* **14,** No. 12, 242–249 (February 1993).
12. K. D. Boese and A. B. Kahng, "Zero-Skew Clock Routing Trees with Minimum Wirelength," *Proceedings of the Fifth Annual IEEE International ASIC Conference and Exhibit*, 1992, pp. 17–21.
13. S. Pullela, N. Menezes, J. Omar, and L. T. Pillage, "Skew and Delay Optimization for Reliable Buffered Clock Trees," *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, 1993, pp. 556–562.
14. R. Kent Dybvig, *The Scheme Programming Language*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1987.
15. C. W. Koburger III, W. F. Clark, J. W. Adkisson, E. Adler, P. E. Bakeman, A. S. Bergendahl, A. B. Botula, W. Chang, B. Davari, J. H. Givens, H. H. Hansen, S. J. Holmes, D. V. Horak, C. H. Lam, J. B. Lasky, S. E. Luce, R. W. Mann, G. L. Miles, J. S. Nakos, E. J. Nowak, G. Shahidi, Y. Taur, F. R. White, and M. R. Wordeman, "A Half-Micron CMOS Logic Generation," *IBM J. Res. Develop.* **39,** No. 1/2, 215–227 (January/March 1995).
16. P. C. Elmendorf, "KWIRE: A Multiple-Technology, User-Reconfigurable Wiring Tool for VLSI," *IBM J. Res. Develop.* **28,** No. 5, 603–612 (September 1984).

Received October 23, 1995; accepted for publication February 12, 1996

David J. Hathaway *IBM Microelectronics Division, Burlington facility, Essex Junction, Vermont 05452 (david_hathaway@vnet.ibm.com).* Mr. Hathaway received the A.B. degree in physics and engineering sciences in 1978, and the B.E. degree in 1979 from Dartmouth College. In 1982 he received the M.E. degree from the University of California at Berkeley. In 1980 and 1981 he worked on digital hardware design at Ampex Corporation in Redwood City, CA. Mr. Hathaway joined IBM in 1981 at the Essex Junction development laboratory, where he is currently a senior engineer. From 1981 to 1990 he was involved in logic synthesis development, first with the IBM Logic Transformation System and later with the IBM Logic Synthesis System. From 1990 to 1993 he led the development of an incremental static timing analysis tool, and since 1993 has been working on clock optimization programs. Mr. Hathaway has three patents issued and seven pending in the U.S., and four publications. He is a member of the IEEE and the ACM.

Rafik R. Habra *IBM Microelectronics Division, East Fishkill facility, Route 52, Hopewell Junction, New York 12533 (habra@vnet.ibm.com).* Mr. Habra received his B.S. and M.S. degrees in electrical engineering, both from Columbia University, in 1966 and 1967. He joined IBM in 1967 in the then Components Division in East Fishkill; he is currently employed there as a senior engineer. He worked first on numerical analysis applications, but soon joined the design automation effort at IBM, still in its early stages during that period. Mr. Habra led an effort to provide a chip design system comprising technology development, manual placement, and wiring, as well as shapes generation and checking. This was used for chip production during the seventies. He then became involved with providing a graphic solution to the task of embedding and checking overflow wires that proved instrumental in shortening the design cycle of chips and TCM modules. Mr. Habra holds a patent on parallel interactive wiring; a second patent on parallel automatic wiring is pending.

Erich C. Schanzenbach *IBM Microelectronics Division, East Fishkill facility, Route 52, Hopewell Junction, New York 12533 (SCHANZEN at FSHVMFK1).* Mr. Schanzenbach received a B.S. degree in physics in 1979 from Clarkson University. He joined IBM Corporation in 1980 at the East Fishkill facility, where he is currently an advisory engineer. In 1980 and 1981, he worked on chip placement, and has spent the last fifteen years developing chip routing tools. Mr. Schanzenbach has one U.S. patent pending and one previous publication.

Sara J. Rothman *IBM Microelectronics Division, East Fishkill facility, Route 52, Hopewell Junction, New York 12533 (rothman@vnet.ibm.com).* Ms. Rothman received the A.B. degree in mathematics in 1974 from Brown University, and the M.A. degree in mathematics from the University of Michigan in 1975. She completed her doctoral course work and taught at the University of Michigan until 1980, when she joined the IBM Corporation. Her first assignment, as part of the Engineering Design Systems organization, was to see whether the brand-new technique of simulated annealing could be used for industrial chip design; since then, she has worked on chip placement.