Part 2: The Layout Problem

Progress Toward the "Ideal" Silicon Compiler

Jerry Werner, Editor-in-Chief

Although conceptually straightforward, the automatic transformation of an electronic system functional specification into a completely laid out integrated circuit is a formidable challenge. The term silicon compiler *is often used to describe the set of tools that would (theoretically, at least) accomplish this task. The first part of the silicon compilation problem, including such tasks as logic synthesis (what one Japanese company calls "gate generation"), was discussed in last month's report (September 1983). In this article, we'll explore recent efforts to automate the "back end," or layout portion, of silicon compilation.*

In one sense, the problem of converting a structural circuit specification to an IC layout has been solved. Several semiconductor suppliers, as well as larger system companies, can automatically or semi-automatically lay out a gate array or a standard-cell chip from such an input specification. However, work continues on the development of more powerful and/or more general tools that will completely automate the implementation-specific portion of the silicon compiler (see Figure 1).

There is a clear distinction between automatic layout techniques for semicustom and cell-based ICs, and what most companies call silicon compilers. In the former case, the functional elements that are placed around the chip (typically either at the gate or MSI complexity level) are pre-designed and characterized, then simply called from a library. In the latter case, however, these functional elements (which can be up to LSI complexity) are designed *as required*. In a sense, the automatic layout of gate arrays or standard-cell ICs is analagous to the design of a printed circuit board with pre-designed (standard) parts, whereas IC layout using a silicon compiler is similar to p.c.b. design with *custom* elements.

Compiling Gate Arrays

Although traditional gate-array automatic-layout techniques do not fall under most experts' definition of silicon compilation, one approach by Lattice Logic Ltd. (Edinburgh, Scotland) is a notable exception. Using Lattice Logic's design tools, users specify designs in a hierarchical structural language called MODEL (Gray *et al.* 1983). A MODEL compiler then creates an intermediate description

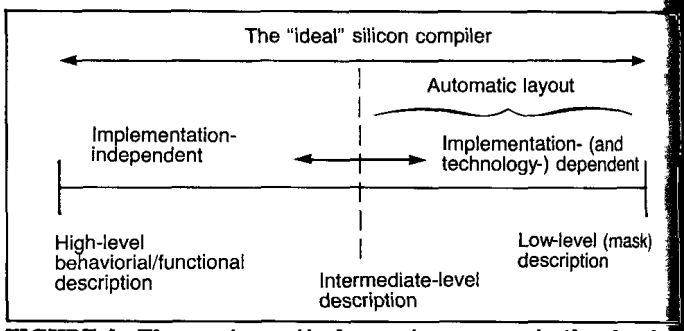


FIGURE 1. The automatic layout process is the *back-end* portion of the ideal silicon compiler.

that is fed in parallel to physical design, simulation a test generation subsystems (Gray *et al.* 1982).

Lattice Logic's system is called a silicon compiler, b cause its cell library is "soft"—that is, the user calls computer programs that describe how the basic cel (consisting of two inverters) are connected to create t desired function. One advantage of creating cells "on t fly" is that unused circuits are removed. For example, the $\overline{Q}$ output of a D-type flip-flop is not used in the circu its associated inverter is eliminated from the cell.

Unlike most commercially-available gate-array automa layout software, Lattice Logic's software cannot easily adapted to a variety of gate array images (generic layout Instead, the LL software actually generates a speci underlying array structure. "That's the nature of compiler—it knows intimately about a design style," co tends John Gray, president of Lattice Logic. "Essentiall we give the gate array away free with our software," h adds. The software creates the base array from a proce design file that contains all the dimensional design rule Gray claims that the software has been successful with wide range of CMOS processes—from 5 micron p-well 3 micron n-well CMOS.

Lattice Logic's software can be used in two related b different modes. In the normal gate-array mode, t underlying array can be pre-designed and pre-diffused with metal personalization added later. In the secon mode, and in a fashion resembling standard cell design, "optimizing compiler" designs an array from the grou up. In this way, chip area is minimized, because the cha nels are only as wide as required by the circuit.

Gray says that the run time for Lattice Logic's softwa

ARAMETERS

ll Specific Parameters

ARAMETER ME	PARAMETER DESCRIPTION	DEFAULT	RANGE Min/Max
A_FILE	PLA code file name	--	n/a
A_TYPE	Specifies PLA core type: SLOW => slow speed, small core version. FAST => fast speed, large core version.	SLOW	SLOW/FAST
TPUT_BUFFER_LOCATION	Specifies each location of outputs for "FAST" type PLA. 1 letter for each outputs. T means cell top B means cell bottom. The word "default" makes all outputs be at cell bottom.	default	n/a
CLOCKED_INPUTS	True creates clocked input buffer.	false	n/a
CLOCKED_OUTPUTS	True creates clocked output buffer.	false	n/a
VSS_WIDTH	VSS bus width for AND plane, OR plane, input and output buffers	12	8/50
VDD_WIDTH	VDD bus width for pullups	12	8/50
INBUFHS_A_PULLDOWN_WIDTH	Gate width for input buffer drivers, ("FAST" core only)	10	8/14
INBUFSS_A_PULLDOWN_WIDTH	Gate A pulldown width in "SLOW" core's input buffer	17	10/20
INBUFSS_A_PULLUP_WIDTH	Gate A pullup width in "SLOW" core's input buffer	5	4/6
INBUFSS_A_PULLUP_LENGTH	Gate A pullup length in "SLOW" core's input buffer	4	4/6
INBUFSS_B_PULLDOWN_WIDTH	Gate B pulldown width in "SLOW" core's input buffer	16	14/20
INBUFSS_B_PULLUP_WIDTH	Gate B pullup width in "SLOW" core's input buffer	5	4/6
EFRESH_NUMBER	Specifies how many minterms and outputs are created before a VSS refresh occurs	10	2

BLE 1. Partial list of the user-defined parameters VTI's PLA "cell compiler." (Dimensions are in mbda, unless otherwise noted.)

a DEC VAX 11/750 computer is approximately 20 ges (40 transistors) per cpu second for the MODEL mpiler, and 2 stages (4 transistors) per cpu second for physical design subsystem. Gray acknowledges that the sical design time is "slightly worse than linear" in reon to the number of stages, but he insists that a ctured design methodology can give a "fantastic comssion in run times—like 60 minutes of cpu time for a 0-gate gate array." Gray explains that "structuring a gn" means partitioning it into logically distinct mod-. This partitioning determines the placement strategy of layout program. "The key concept is using the engi-'s ideas about architecture in placement," he adds.

mpiling Pieces of a VLSI Chip

n contrast to initial silicon compilation efforts in the 1970s, the current emphasis is on compilation at the dule level, rather than at the whole-chip level—specially for production-quality devices. One of the commercial firms to propose a methodology and tools upport block compilations is VLSI Technology, Inc. n Jose, CA). VTI supports what it terms "cell mpilers"—programs that will automatically generate uts of specific functions ranging from simple gates to e complex functions, such as PLAs and multi-ers/demultiplexers.

lthough VTI uses the term *compiler*, at least one indi-al contends that they are taking undue liberties with the word. "VTI is making a big hoopla about 'compilers' that are, in reality, parameterized cells," says Jeff Siskind, president of a fledgling silicon compiler company called MetaLogic, Inc. (Bedford, MA). Siskind was formerly the principal researcher on the MacPitts silicon compiler project at MIT's Lincoln Laboratories (Siskind *et al.* 1982). "Anybody who has a layout language embedded in another high-level language has parameterized cells," he adds.

Indeed, a user of VTI's cell compilers specifies physical size or electrical *parameters*, upon prompting by the software. Alternatively, the user can use the *default* parameters pre-entered into the cell compiler program. Table 1 shows the list of parameters for VTI's HPLA32 static programmable logic array (PLA). This list defines a PLA cell (maximum access time of 120 ns) with up to 25 inputs, 30 minterms and 30 outputs. Larger PLAs, with correspondingly longer access times, can also be laid out.

At the time of this writing, VTI offered two different cell compiler libraries (HMOS only): the VTS 810 "primary" library and the VTS 820 "extended" library. The former includes simple buffers, gates, latches, and flip-flops, and is routinely supplied with the other design software. In addition to the PLA generator described above, the VTS 820 library includes (among others) counter, adder, ROM, RAM, and ALU cell compilers. VTI requires a fabrication commitment from users before it will release the extended cell-compiler library.

In-Tune Texans

Even Texas Instruments, which in recent months has backed off from several of its earlier announced plans for semicustom (gate-array) devices, is now talking about offering parameterized cells. At a recent press conference called to unveil the company's new SN54/74SC standard cell library and design approach (see "News" in this issue), one slide labeled "The Future" made the following statement: "Procedural cells will be added, providing some unique new capabilities . . . RAM, ROM, [and] N-wide counters, registers, etc." (A TI spokesman confided that the company had not yet settled on a suitable approach for customer specification of such cells, but he hoped the problem would be solved by the end of 1983.)

Compiling Blocks for DSP Chips

Layout programs that "compile" portions of a chip layout are emerging for data processing and random logic applications, and for digital signal processing (DSP) applications. At the recent Custom Integrated Circuits Conference, one researcher from the Philips Research Laboratories (Eindhoven, The Netherlands) described a layout generator for parameterized array-multipliers (Benschop 1983). Philips' approach to creating a parameterized array-multiplier layout generator was as follows:

1. The array was logically decomposed into slices that were further decomposed into cells having regular structures.
2. Simulations were performed on a small ("minimal") array containing all the *types* of cells that would likely be contained in a larger array.

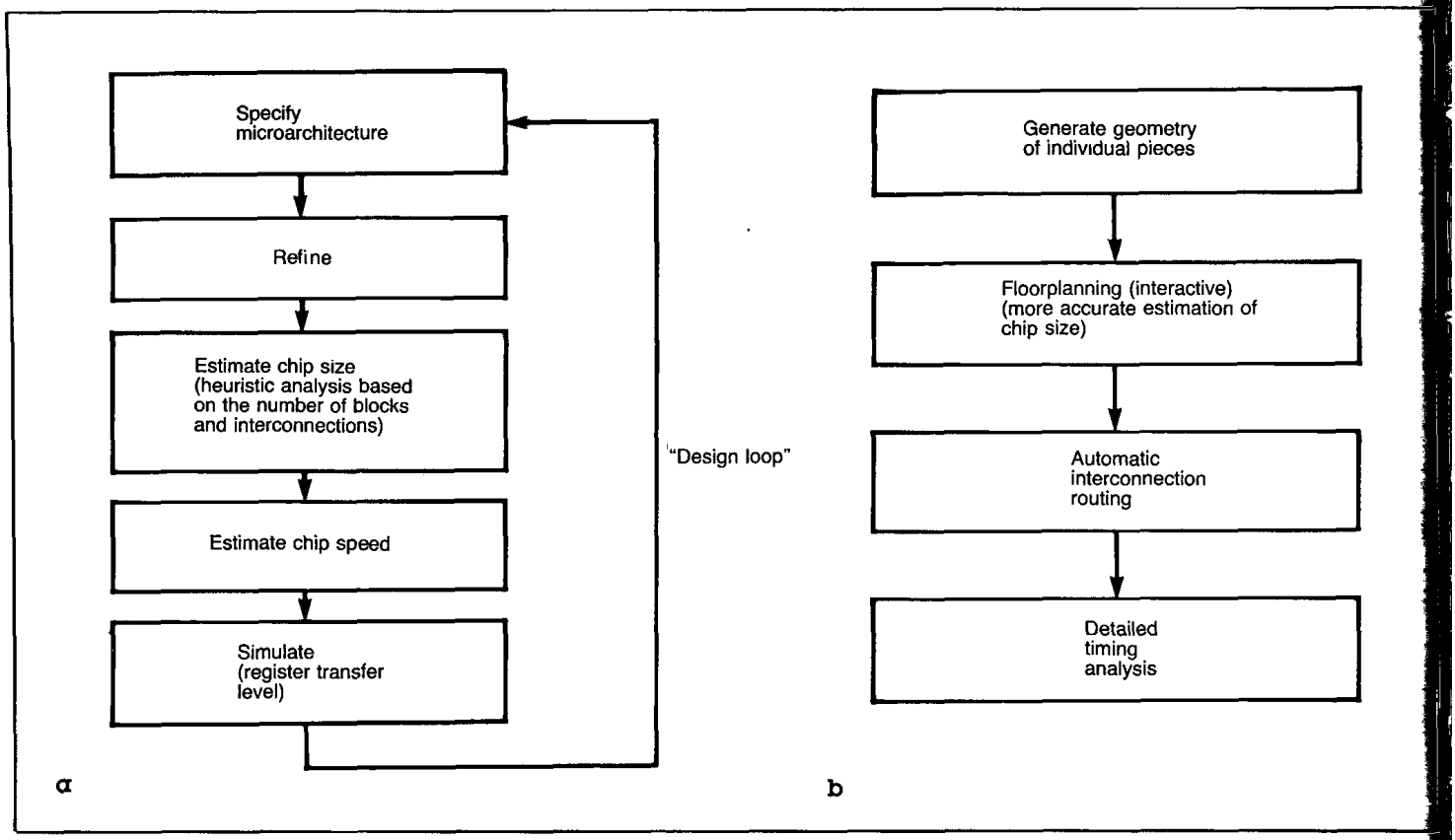


FIGURE 2. According to Silicon Compilers, silicon compilation consists of two parts: a "high-level" design phase (a), and an artwork-generation phase (b).

3. The basic *cell* was carefully laid out so that it could be used in all known contexts. This cell was then simulated under worst-case conditions.
4. Local cell parameters and global composition parameters were built into the layout description.

Perhaps a more ambitious approach to compiling digital signal processing chips is being taken by researchers at the University of Edinburgh in Scotland. They have developed the F.I.R.S.T. (Fast Implementation of Real-time Signal Transforms) silicon compiler (Bergmann 1983), which lays out a network of interconnected bit-serial operators on a relatively fixed floorplan.

Like Lattice Logic's approach, the F.I.R.S.T. software has a structural description language that describes the function blocks and their interconnection. (The similarity to Lattice Logic's MODEL language is not a coincidence; most of Lattice Logic's founders came from the University of Edinburgh.) The F.I.R.S.T. tool set also includes two simulators: one that uses data extracted from a chip layout, and another that simulates operators at a higher level.

Higher Level Tools

The use of functional (or register-transfer level simulators) in conjunction with module generators is gaining popularity. Two-year-old Silicon Compilers, Inc. (Los Gatos, CA), many of whose founders came from Caltech's now defunct Silicon Structures Project (SSP), is a case in point. According to SCI president Phil Kaufman, system designers will need tools to specify and "compile" ICs at the architectural level, instead of at the transistor level. Kaufman says that future system designers "will not need to lay out MOS transistors to design VLSI circuits."

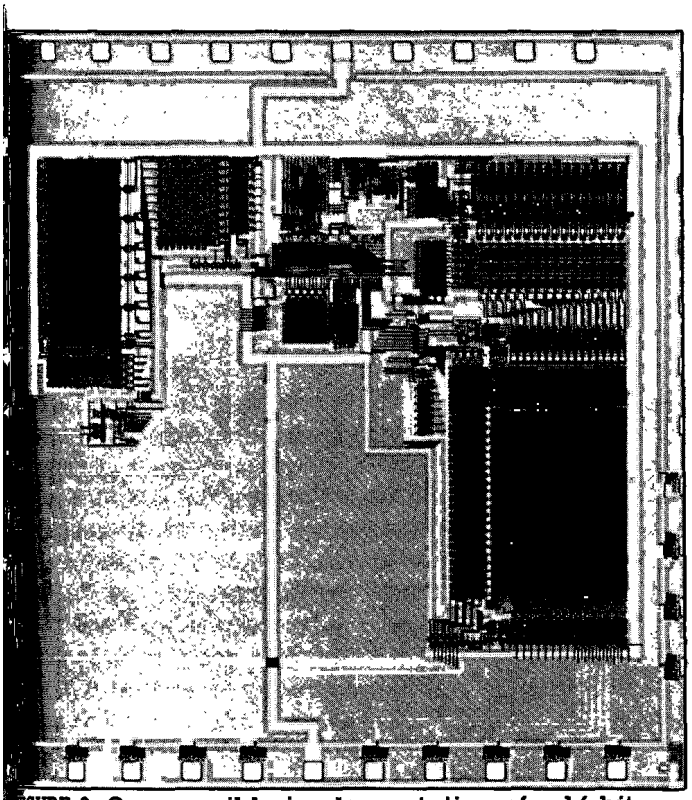
Kaufman is reluctant to divulge specifics about SCI's suite of tools under development, but says it will include a high-level functional simulator, block generators for laying out specific functions (an approach SCI terms "compiled geometry"), and tools for automatically laying out and interconnecting those blocks (see Figure 2). "For us, you start and finish at the block diagram level," he explains.

SCI's approach appears to be nearly identical to that of nearby VLSI Technology, Inc.; however, Kaufman contends that "they're [VTI] coming up from the bottom and we're coming down from the top." Like VTI, SCI is developing parameterized cell generators, but Kaufman emphasizes that parameters required for SCI's generators will primarily be performance related, whereas VTI's parameters are at a lower level (*e.g.*, gate lengths).

One of SCI's founders is Dave Johannsen, whose Ph.D. thesis at Caltech described the "Bristle Blocks" silicon compiler (Johannsen 1979). Although Bristle Blocks is often cited as one of the pioneering attempts at silicon compilation, Kaufman says that "Bristle Blocks had several limitations that prevented it from being suitable for commercial chip designs." These limitations, he explains, included its preconceived notion of a chip's internal architecture, and its inability to concatenate fairly rigid blocks consistently.

The Plex Project

Although SCI now eschews any approach to silicon compilation requiring a preconceived notion about the chip architecture, others continue to pursue just such a tack. At Bell Labs (Murray Hill, NJ), researchers recently described



GURE 3. One possible implementation of a 16-bit ex microcomputer. (Photo courtesy of Bell boratories.)

program that automatically generates nMOS microcomter layouts (Buric *et al.* 1983). Although these so-called Plex" computers are intended to be components on larger LSI devices, they are individually quite complex: up to -bit microcomputers have been tested (see Figure 3).

A primary driving force behind Bell Labs' Plex project the desire to give the chip designer an easy-to-specify gh-level building block. The user needs only to specify *program* (in assembly or C language) that the microputer is to execute. A conventional software compiler termines which instructions, from a global instruction , are actually used in that program. This selection termines the microcomputer's specifications, including number of registers needed in the data path, and the e and contents of the data ROM. Once an intermediatevel representation of the microcomputer has been nerated, a hierarchy of special macrocell layout nerators creates the layout. (In such a "hierarchy," mod- generators can call lower-level module generators until most primitive level is reached.)

Misha Buric, head of the Interactive Digital Systems Rearch Department at Bell Labs, and leader of the threean Plex effort, is quick to point out that the Plex proach is still experimental.

Buric and his associates Tom Matheson and Carl Chrisasen at Bell Labs have developed cell generators that e into account both electrical specifications and physical e requirements (Matheson *et al.* 1983). The ability to nge the aspect ratio of compiled cells suggests that it's sible to create blocks automatically that can be tailored specific area and connection constraints imposed by the erall chip layout plan.

Indeed, other researchers are working to develop ways generate cells with specific dimensions. At the University of Southern California, a team headed by Professor Mel Breuer is studying what is being termed "row-column (or *RC*) synthesis" of IC functional blocks (Breuer and Knapp 1983). Specifically, Breuer's group is studying *vector functions* that can be decomposed into arrays of identical or near-identical cells. While this work appears to be in its infancy, one potentially far-reaching finding has emerged: the aspect (length-to-width) ratio of the kernel cells has little or no effect on the aspect ratio of the overall functional block.

Merging Blocks Into a Chip

The increasing use of module generators has simultaneously created a need for better techniques to compose a complete chip layout from variably-shaped blocks. Better floorplanning techniques that can be used *before* the blocks are compiled may be part of the solution to this problem. The results of the floorplanning exercise could then be used as inputs to module-generation programs that take physical constraints into account. Theoretical work at IBM (Heller *et al.* 1982) and at the University of Edinburgh (Brebner and Buchanan 1983) is addressing the variable-size-block floorplanning problem. Indeed, this work may help solve one of the more difficult problems still facing developers of the "ideal" silicon compiler. □

References

Benschop, N.F. May 1983. "Layout Compiler for Variable Array-Multipliers," *Proceedings of the Custom Integrated Circuits Conference*, Rochester NY.

Bergmann, N. 1983. "A Case Study of the F.I.R.S.T. Silicon Compiler," *Third Caltech Conference on Very Large Scale Integration.*

Brebner, G. and D. Buchanan. September 1983. "On Compiling Structural Descriptions to Floorplans," *Proceedings of the IEEE International Conference on Computer-Aided Design*, Santa Clara, CA.

Breuer, M.A. and D.W. Knapp. 1983. "Row-Column Synthesis of VLSI Macrocells," *Proceedings of the International Symposium on Circuits and Systems (ISCAS)*, Newport Beach, CA.

Buric, M.R., C. Christensen, and T.G. Matheson. September 1983. "The Plex Project: VLSI Layouts of Microcomputers Generated by a Computer Program," *IEEE International Conference on Computer-Aided Design*, Santa Clara, CA.

Gray, J.P., I. Buchanan, and P. Robertson. 1982. "Designing Gate Arrays Using a Silicon Compiler," *Proceedings of the 19th Design Automation Conference*, Las Vegas, NV.

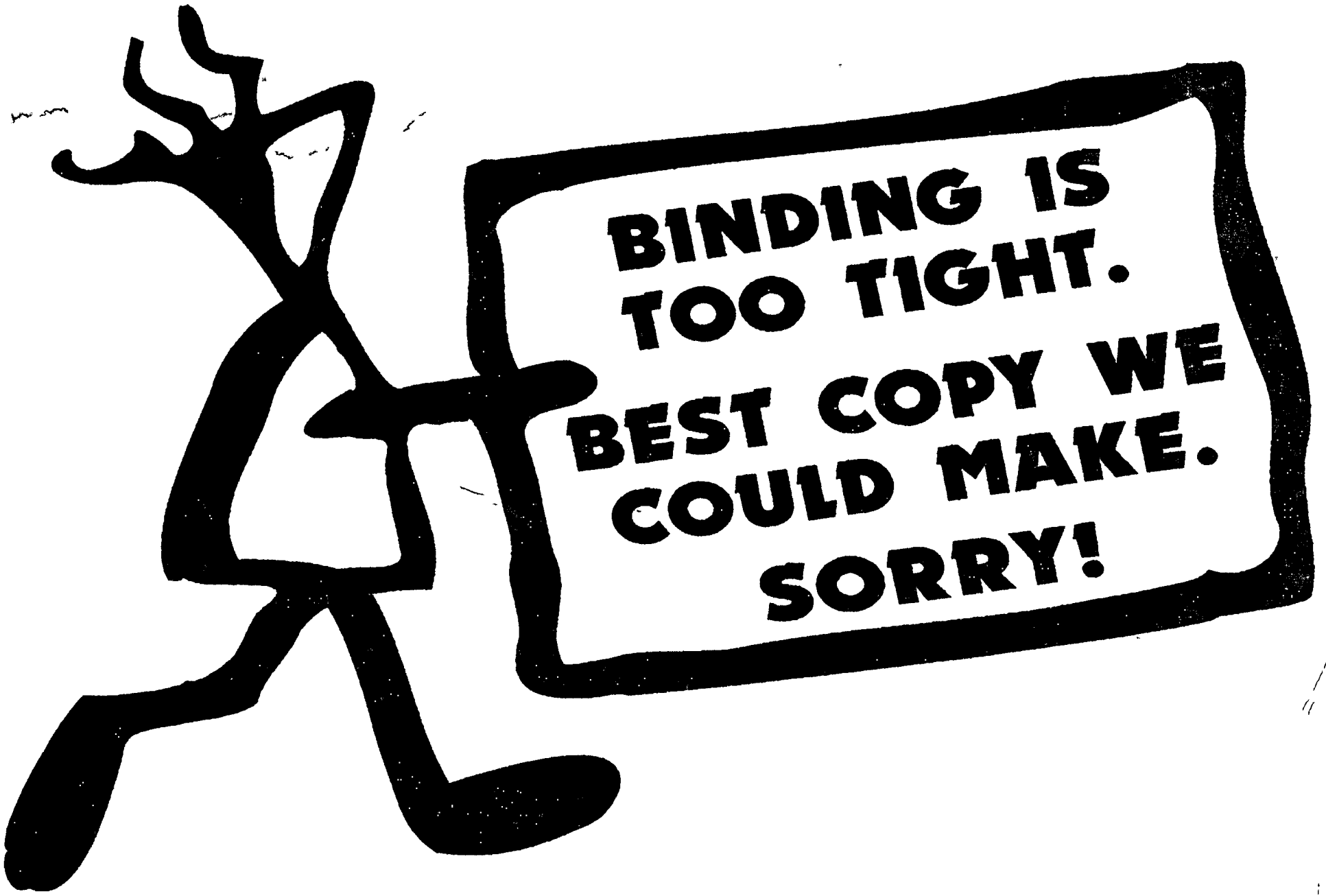
Gray, J., I. Buchanan, and P. Robertson. October 1983. "Controlling VLSI Complexity Using A High-Level Language for Design Description," *International Conference on Computer Design*, Port Chester, NY.

Heller, W.R., G. Sorkin, and K. Maling. 1982. "The Planar Package Planner for System Designers," *Proceedings of the 19th Design Automation Conference*, Las Vegas, NV.

Johannsen, D. 1979. "Bristle Blocks: A Silicon Compiler," *Proceedings of the 16th Design Automation Conference.*

Matheson,T.G., M.R. Buric, and C. Christensen. September 1983. "Embedding Electrical and Geometric Constraints in Hierarchical Circuit-Layout Generators," *International Conference on Computer-Aided Design*, Santa Clara, CA.

Siskind, J.M., J.R. Southard, and K.W. Crouch. 1982. "Generating Custom High Performance VLSI Designs From Succinct Algorithmic Descriptions," *MIT Conference on Advanced Research in VLSI*, Cambridge, MA.


BINDING IS
TOO TIGHT.
BEST COPY WE
COULD MAKE.
SORRY!