Borrower: IPL(OCLC)

Lending String:
EXH,*ZXC,OIT,CDS,RVE,TXH,WAU,PMC,CPO,VPI,IWA,
KLG,AZS,INT

Journal Title: Digital design.

Volume: **Issue:**
Month/Year: August 1984 **Pages:**

Article Title: Silicon compilation; a revolution in VLSI design

Article Author: Collett, R.

Imprint: Boston, [etc.] Benwill Pub. Corp.

Call #:
Location: cohen 1st floor

Shipping Address:
Purdue University Libraries-ILL
504 West State Street
West Lafayette, IN 47907-2058

Max Cost: 65.00IFM

Odyssey: 128.210.126.171
Ariel: 128.210.125.135 (IF NO EXTRA CHARGE)
Fax:
Email: ill@purdue.edu

ILL Number: 127840778

☐ CONDITIONAL ☐ CHARGE ☐ NOS

Silicon Compilation: A Revolution In VLSI Design

by Ronald Collett, Technical Editor

Until recently, a system that automatically synthesized complex VLSI chips with minimal human intervention was wishful thinking. However, several research laboratories and a few vendors have been quietly engaged in the development of this ultimate IC design tool—a silicon compiler. Not surprisingly, silicon compilation promises to have dramatic impact on the entire electronics industry.

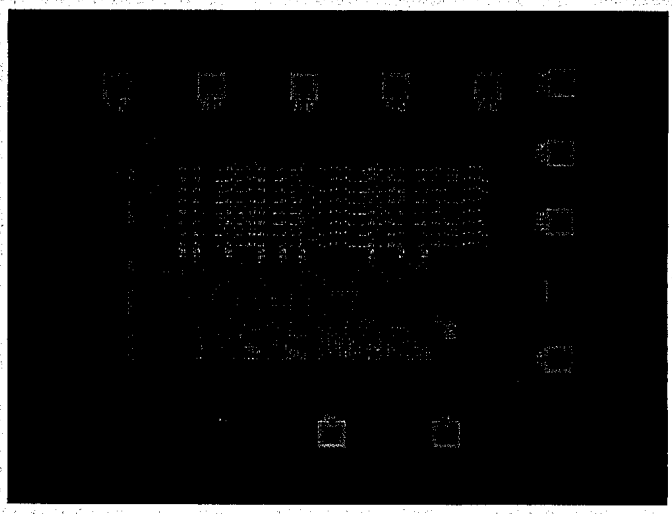
To fully appreciate the power of a silicon compiler, compare it to a Fortran compiler and the difference between writing a program in Fortran and writing that same program in machine language. With the Fortran program, the compiler converts the high level instructions to machine language. Similarly, a system architect describes the operation of the chip in a high level language and the silicon compiler automatically synthesizes the masks necessary to fabricate the chip.

The expanding application-specific integrated circuit market has spawned a new generation of tools that automate VLSI design.

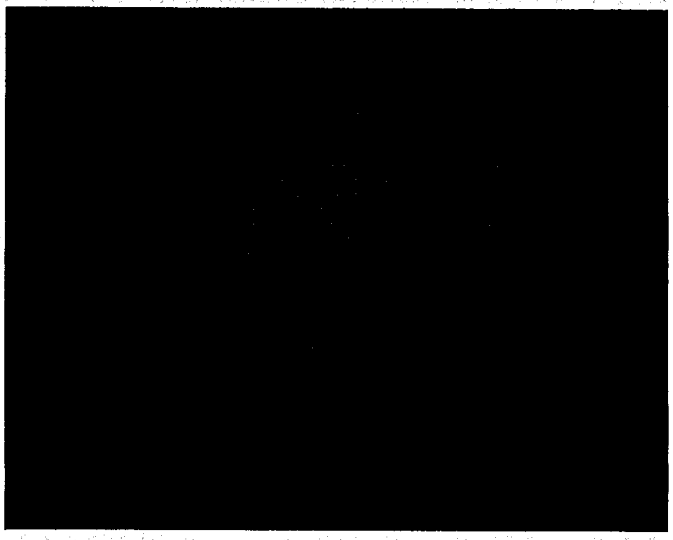
Forces Driving Silicon Compilation

Handcrafted full-custom chips typically require one to two years to develop, with costs ranging from $200,000 to $1 million. In addition, only a few highly specialized engineers have the skills necessary to design a full-custom chip. To make matters worse, high volume and a wide market window are two criteria for the profitability of a handcrafted IC. If these parameters are questionable, the development of a custom chip is often too risky.

Lengthy design cycles, high costs and shortages of chip designers have given impetus to other methodologies that facilitate the design of application-specific integrated circuits (ASIC). Among the various ASIC alternatives, semi-custom ICs (i.e. gate arrays and standard cells)



Left: This 16-bit, "Fanatically Reduced Instruction Set Computer," (FRISC) was designed by Metalogic's silicon compiler. Above: This simplified Universal Asynchronous Receiver/Transmitter (UART) was also designed with Metalogic's silicon compiler.



Using Metalogic's silicon compiler, this Automatic Gain Control (AGC) chip was designed in less than three weeks by a digital systems designer who had no previous IC design experience.

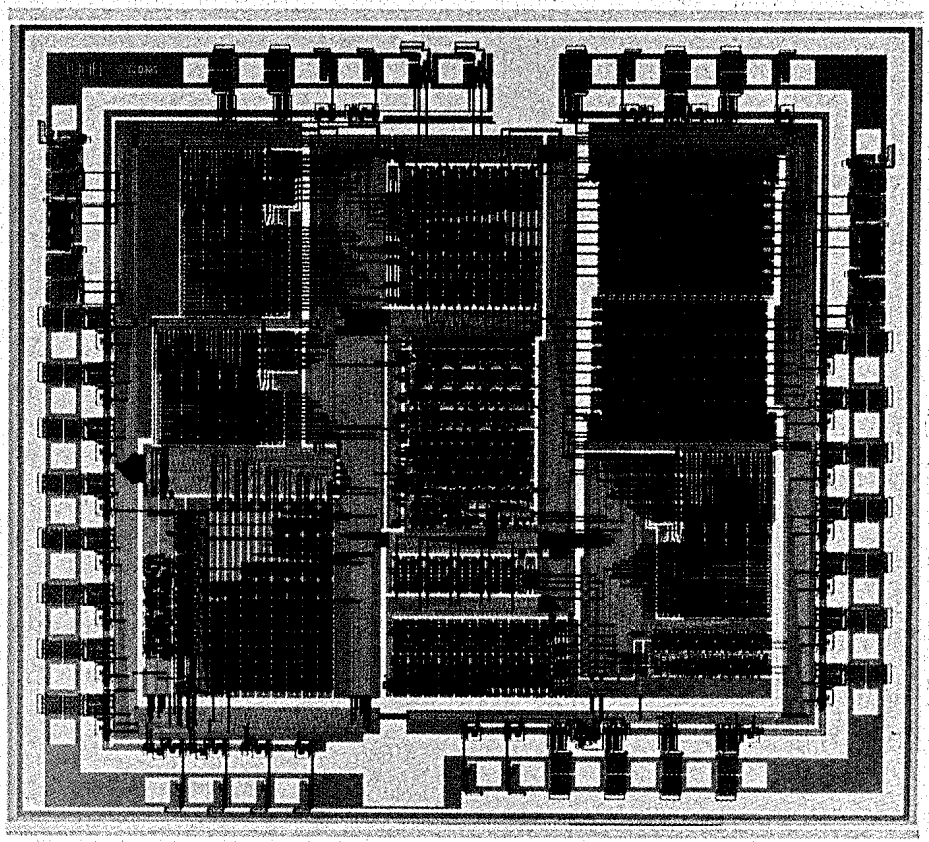
are spearheading the migration away from off-the-shelf components. Unlike full-custom, semi-custom and silicon compilation allow logic designers to use their current skills to build chips. The latter is based on a different design philosophy and until now, has been confined to the research lab.

Absolving the design engineer from transistor layout tasks is the primary benefit of a semi-custom chip. For instance, imagine a particular design having 25 blocks (from a block diagram) which corresponds to 5000 gates, translating to 12,000 transistors—equaling nearly 250,000 polygons. With semi-custom, the design engineer would be responsible for managing and implementing only the block and gate level data. In addition, turn around times for these parts typically span six to 20 weeks with non-recurring engineering (NRE) costs ranging from $5,000 to $75,000. In comparison to full-custom handcrafted designs, there is little doubt that gate arrays and standard cells have overwhelming advantages. However, semi-custom design forces the system architect to become entangled in circuit and logic design.

Aside from being caught in a labyrinth of transistors, gates or registers, traditional design routes force engineers to follow an inefficient design cycle. Although most projects are organized in an efficient top-down style on paper, those models are not a reflection of actual design practices.

In a typical design cycle, for example, the system problem is defined first, then the system definition is born. Drawing a block diagram of the solution is the next step and each engineer on the project is given a portion of the system to implement (**Figure 1a**). Observing this common style of organization and flow, one would think that this is a true top-down procedure. However, since each engineer is designing a small piece of the system and rarely knows how the entire system works, the design is actually being implemented from the bottom up (**Figure 1b**).

The engineers designing the logic are usually given a specification of the electrical output signals their circuit is ex-



Designed by Silicon Compilers Inc. and marketed by Seeq Technologies, this Ethernet data-link controller chip was the industry's first Ethernet local area network VLSI product.

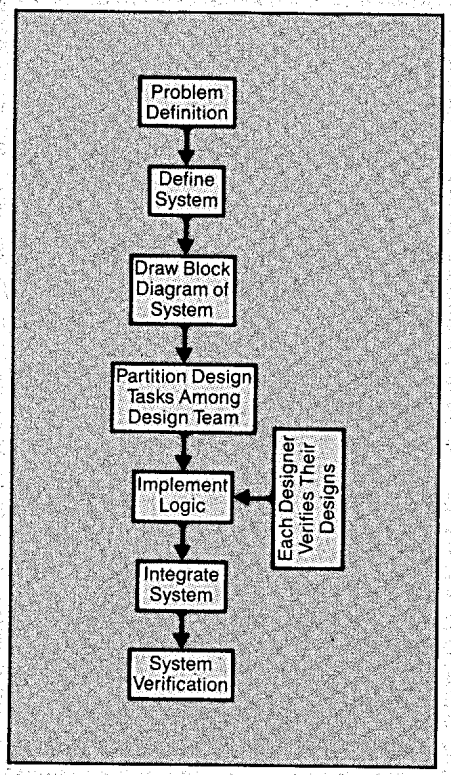


Figure 1a: This flow chart outlines a typical design cycle using a top-down approach. This is, however, only a conceptual procedure and not a reflection of what actually occurs.

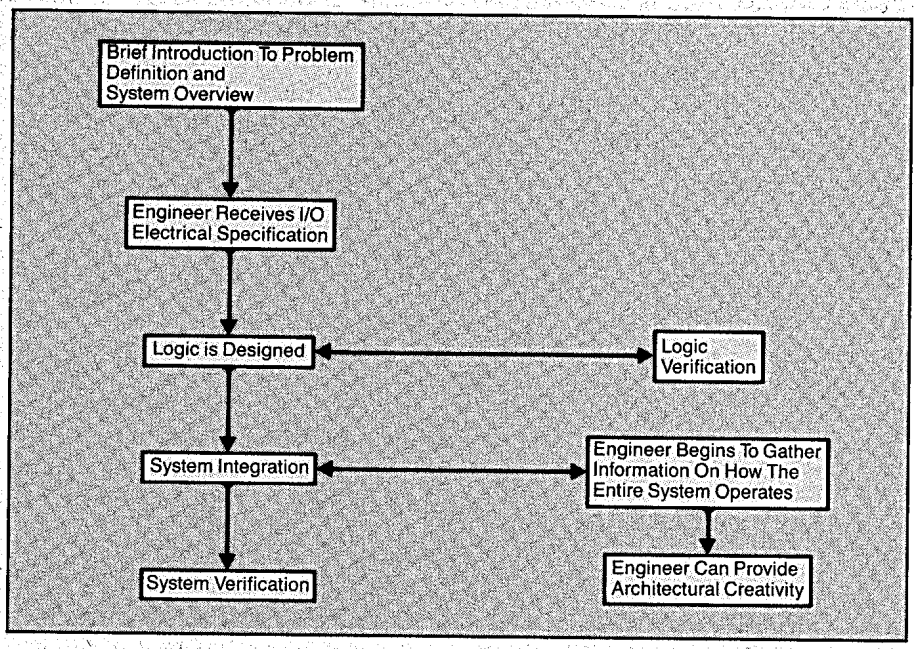


Figure 1b: Each engineer actually implementing the circuitry follows this design cycle. Compared to Figure 1a, this is a bottom-up procedure which discourages the engineer from exercising architectural creativity until the latter stages of the design cycle.

pected to produce. This is based on another specification outlining the input signals which are coming from another engineer's design. As a result, the building blocks that make up a complete system are put together by many individuals who typically have limited knowledge of how these elements are related to the remainder of the system.

With this scenario in mind, the design is orchestrated using a top-down strategy and then implemented from the bottom up. From a system point of view, the engineers' creativity is never realized since they are not given a full appreciation of global architecture. Only the few that conceived the architectural definition are able to exercise creativity at the system level.

In addition, once the logic is designed, it is very difficult and excessively time consuming to alter the system's architecture. This discourages engineers from making architectural enhancements—even if they recognize areas needing improvement. Freeing the designer from this constraint is one of the primary advantages of silicon compilation.

In contrast to semi- and full-custom designs, silicon compilation divorces the system architect from transistor layout and logic/circuit design. With this portion of the design cycle eliminated, engineers can spend more time on optimizing the chip's architecture. Of equal importance, silicon compilation provides greater leverage with the minimum effort necessary to achieve maximum output. Gaining maximum leverage is a primary concern when choosing among the various VLSI design methodologies and tools. With this in mind, silicon compilers have the potential to decrease the design cycle's length by an order of magnitude.

Approaches to Silicon Compilation

Presently, there are two schools of thought on how silicon compilers should tackle design problems. These differences of opinion boil down to a single argument—how much of the design should be done by the compiler versus how much should be performed by the human. Compilers from one camp rely on engineers to define the chip's architecture, whereas the second school allows the computer to take on those responsibilities.

The first group maintains that silicon compilers cannot adequately solve the architectural problems associated with designing a VLSI chip. After the design team delineates the chip's architecture, the compiler is responsible for implementing the various functions that have been outlined. Vendors including Silicon Compilers Inc. (SCI) (Los Gatos, CA) and VLSI Technology Inc. (VTI) (San Jose, CA) have taken this approach.

VLSI Technology's system is based on an Apollo computer and includes 175 different "cell compilers." Some of the more sophisticated compilers synthesize such functions as PLAs, ROMs, RAMs and ALUs. Selecting a particular cell compiler from the library causes the system to prompt the user for electrical parameters that are variable within the cell. With these parameters, the compiler generates the geometrical primitives of an IC layout in the Caltech Intermediate Format.

On the other hand, according to SCI's President, Phil Kaufman, the turnkey system SCI is planning to introduce next month (September, 1984) will be much different from VTI's system. It appears that SCI's system will require less detail about the function it's told to implement, and rely more on register transfer type data. Kaufman explains, "Our silicon compiler supports exploratory design. In other words, given a little bit of input information about the architecture, the compiler will provide the engineer with parameters such as die size, speed and power consumption."

SCI's silicon compiler is based on David Johannsen's (one of the firm's founders) doctoral thesis on silicon compilation prepared at the California Institute of Technology in 1981. Bristle

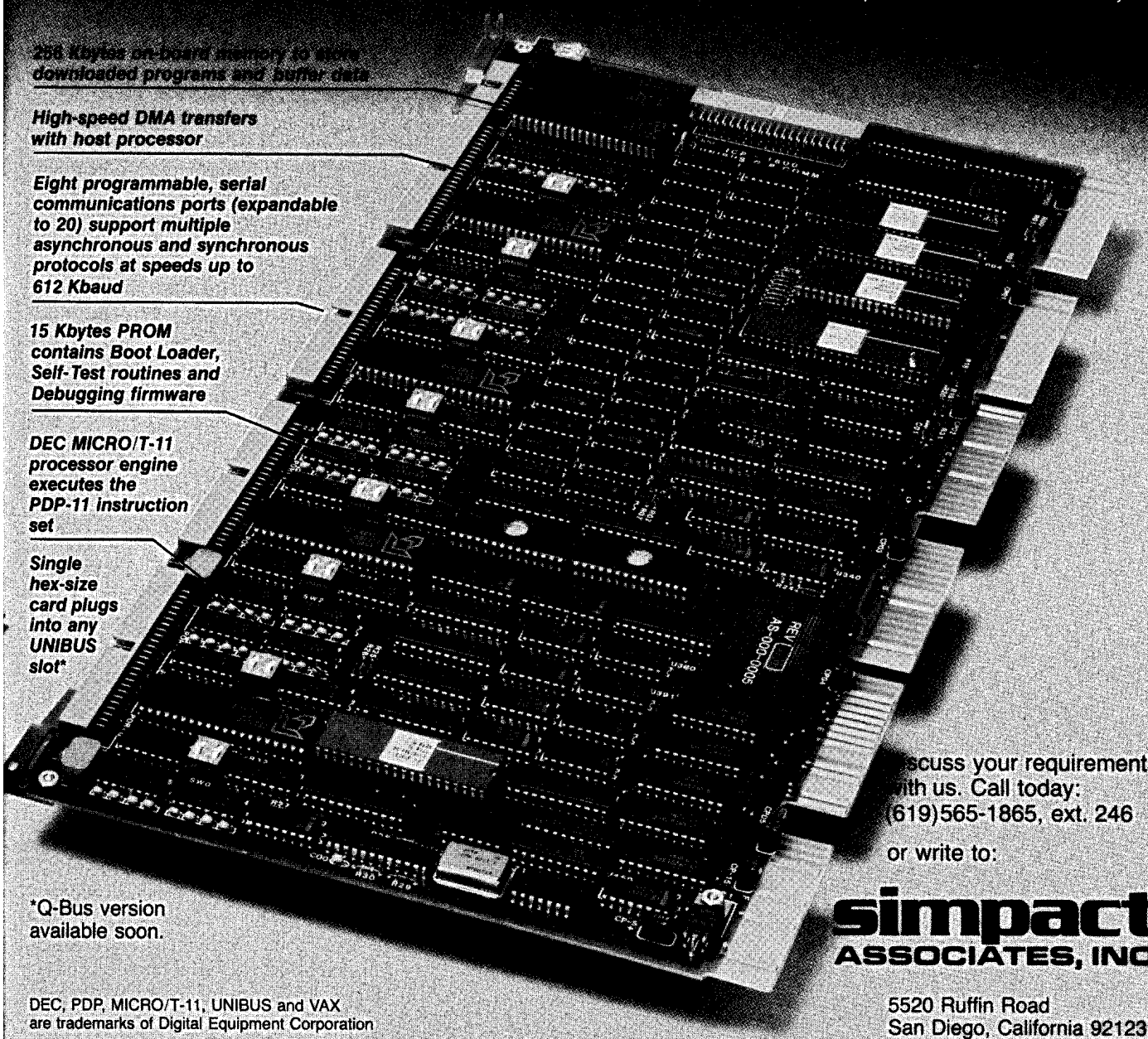
At Last!
A Communications Processor For DEC Computers That You Can Program

The Simpact ICP1600 Intelligent Communications Processor can off-load your host system because it's versatile enough to handle the toughest communications tasks and is code compatible with VAX and PDP-11s.

Use your host system utilities and our software toolkit to create your custom protocols, or let our communications specialists help you develop them.

X.25 software is available from Simpact now, and more standard packages are on the way.

For under $5,000, the ICP1600 is the most cost-effective communications processor available today.



*Q-Bus version available soon.

DEC, PDP, MICRO/T-11, UNIBUS and VAX are trademarks of Digital Equipment Corporation

scuss your requirements
ith us. Call today:
(619)565-1865, ext. 246
or write to:

simpact
ASSOCIATES, INC.

5520 Ruffin Road
San Diego, California 92123

Write 25 on Reader Inquiry Card

Blocks, the name given to Johannsen's compiler, focused on generating geometries for IC layout. It synthesized a structured data flow and was limited in the types of architectures it could build. SCI claims that Bristle Blocks has been extended so that virtually any architecture can now be synthesized.

Analog circuits, however, cannot be synthesized by the compiler. To integrate analog circuits into a chip that the compiler is creating, the user can either build the function or call it up from an already established analog library. Once the electrical and physical parameters of the circuit are known, the user tells the compiler that a prefabricated design is to be included on the chip.

Although SCI's turnkey silicon compilation system will not be formally introduced until next month, the firm has made the compiler's accomplishments known. Three chips have been synthesized, including the data path chip for Digital Equipment Corporation's MicroVAX I (**Figure 2**). This chip was reportedly finished in seven months. However, details about the amount of human effort that went into the chip's development were not made public.

Other achievements from SCI include an Ethernet controller which was designed by Seeq Technology Inc., and a high resolution graphics chip which is used in Sun Microsystem's workstations. The Ethernet and graphics chips were reputedly developed in five months and nine months, respectively.

The Second Approach

For the most part, the second approach to silicon compilation has stayed in the research laboratory. Since this version puts far greater design responsibility in the hands of the compiler, it is much more difficult to develop.

A spin-off from MIT/Lincoln Laboratory, Metalogic (Cambridge, MA) is engaged in the development of a compiler that falls into this second category. While working at MIT/Lincoln Laboratory, the founders of Metalogic, Jeff Siskind, Jay Southard and Ken Crouch developed MacPitts—a silicon compiler funded by the Defense Advanced Research Projects Agency. This combined research effort is the foundation of the firm's current work.

According to Southard, the firm is currently putting the final touches on a silicon compiler which is scheduled to go into beta site this month. The new compiler, called MetaSyn, is an extension of the MacPitts effort.

One of the problems of MacPitts was that it did not have a performance prediction system. Without this, it was very difficult to determine an IC's critical timing areas and maximum clocking speed. Metalogic recognized many of the shortcomings of the MacPitts effort and has since incorporated these enhancements into MetaSyn.

Although their system requires some human intervention, it can supposedly extract register transfer data from a simplified behavioral description of the chip. The behavioral input is in the form of an algorithm which describes how the chip operates. The language used to interface to the compiler is similar to LISP, although it has many characteristics specifically tailored to IC design.

MetaSyn makes architectural decisions and tradeoffs about the hardware used to implement the algorithm. For example, if a macrofunction composed of an accumulator and a multiplier were synthesized by MetaSyn and the input algorithm stated that another multiplication function was needed, MetaSyn would explore whether the multiplier contained in the macrofunction could be used to perform this additional multiplication task. If MetaSyn found that the multiplier contained within the macro (multiplier/accumulator) was not used when the additional multiplication task was needed, it would implement some extra control logic so that the imbedded multiplier could be used during that time period.

As a result of this exploration, the silicon compiler would synthesize simple control logic instead of another complex multiplier. This allows design engineers to explore not only the tradeoffs associated with the hardware implementation but also the performance of the algorithm driving the hardware. This is far more important, since the hardware is only as good as the algorithm.

Silicon Compilation In A Nutshell

The objective of silicon compilation is to transform behavioral or structural descriptions into the geometric data necessary to fabricate an IC. The difference between the two approaches to silicon compilation is determined by whether the description is behavioral or structural. Also, it should be understood that silicon compilation is a concept as opposed to a piece of software. It involves the transformation of a conceptual description to a concrete piece of circuitry. As a result, there are many differing ideas about the exact definition of a silicon compiler.

A complete explanation of the various approaches to silicon compilation is best described in the Y-chart developed by Gajski and Kuhn shown in **Figure 3**. Their objective was to develop a hierarchical model of the various approaches to VLSI design. Each line segment of the Y depicts one of the various ways to describe the functional elements (building blocks) of a chip: behaviorally, structurally or geometrically. All three are then further broken down hierarchically such that by moving from the center to the outside of the Y, the elements become larger.

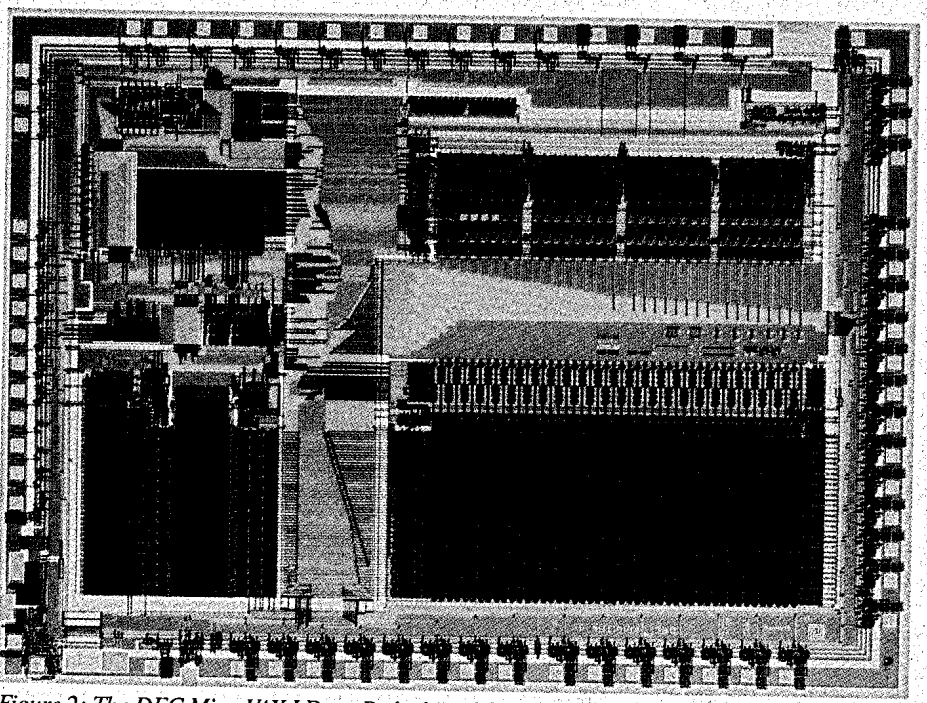


Figure 2: The DEC MicroVAX I Data-Path chip, developed by Silicon Compilers Inc. was finished in seven months.

The dotted arcs shown between the axes represent CAD tools that transform one type of representation to another. For example, the two arcs shown in **Figure 4** depict a silicon compiler that can transform Boolean expressions to gate representations and then to mask geometries. The loop on this gate level represents an analytical tool such as a logic simulator. Similarly, the loop on the mask geometries represents an automatic design rule checker.

Behavioral representations sit at the highest and most abstract level of the Y-chart and are implemented as a function of time. For example, if the system architect wanted a circuit to add two numbers (x and y), the memory location of these variables and the type of ALU would be left to the compiler's discretion. The description would say nothing about how the addition function is implemented. It would simply instruct the compiler to synthesize a device that added two numbers at time=t1 and output the results at time=t2.

At its lowest level, the behavioral characterization might take the form of Boolean equations. Moving away from the center, the Y diagram shows the compiler accepting behavioral data in the form of an algorithm. (Metalogic's silicon compiler, MetaSyn, is represented at this point in **Figure 5**). Finally, at the apex of the behavioral axis, the system architect could describe the chip in terms of its system level input and output behavior.

Moving down the hierarchy (across the Y), a compiler working from a structural description defines a distinct set of functions to implement, say, a microprocessor. With this type of compiler, the chip's global architecture is first defined by the system architect. The various blocks needed to implement the design would then be constructed by the silicon compiler. (SCI's silicon compiler is represented at this position on the Y diagram of **Figure 6**). In this case, the fact that the microprocessor's architecture was defined by the system architect and not by the computer highlights the major difference between structural and behavioral strategies.

Compilation shown on the structural axis accepts only structural data, such as a transistor, gate or register. Such a structural description can come from one of two sources. As described above, the design team may specify the input data. The other possibility is that structural output data from a compiler on the behavioral axis can act as the front end. In this case, compilation that falls along the structural axis is an interim compilation level of behaviorally specified data.

The geometrical representation of the Y diagram ignores the function that the chip is destined to perform and deals primarily with the hardware implementation of the structural or behavioral parameters. Moving away from the Y's hub, the first level would be mask geometries which refer to the polygons used to form a transistor. Continuing along the geometric axis, a cell could pertain to several transistors joined together to make a gate. Layout refers to the physical silicon area where a function is assigned and represents the highest level in geometric hierarchy.

The ideal silicon compiler requires no human intervention to define a chip's architecture. So, to make design tradeoffs, the compiler must have a certain amount of intelligence. Research into the use of expert systems coupled to silicon compilers is underway at the University of Illinois under the direction of Dr. Daniel Gajski. Other research into the use of expert systems has also come from Carnegie-Mellon University. Some of the results of their work include an automated design tool that synthesized an MCS-6502 microcomputer in four hours. As research into this technology continues, these automated chip design tools will acquire greater decision-making power.

Problems To Conquer

Large die sizes are one of the major problems that plague today's silicon compilers. Chips synthesized by these automated tools have yielded dies that are 1.3 to 10 times larger than those done by hand. (The more advanced compilers can consistently produce chips 1.3 to 3 times as large.) With the cost of silicon at a premium, this poses a serious obstacle to industry's acceptance of this new technol-

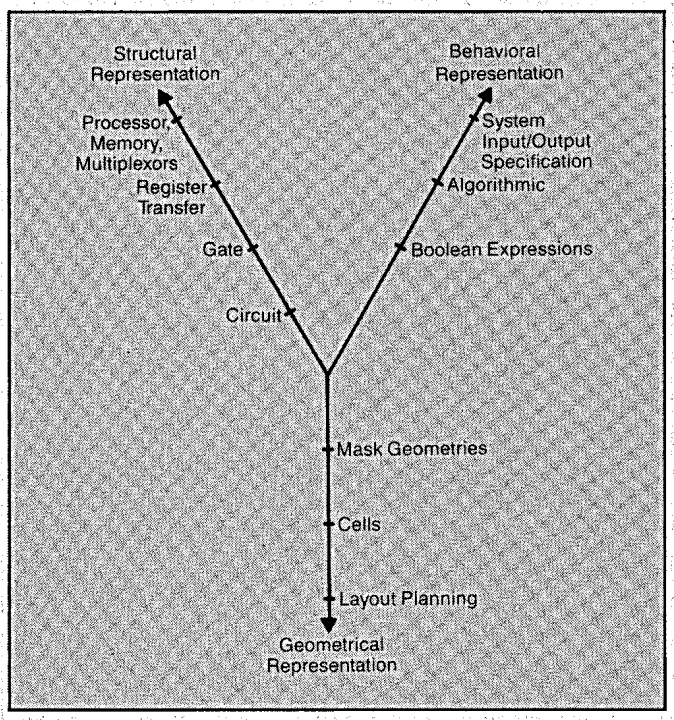


Figure 3: The Y diagram illustrates the three ways to represent the VLSI design data: behaviorally, structurally or geometrically.

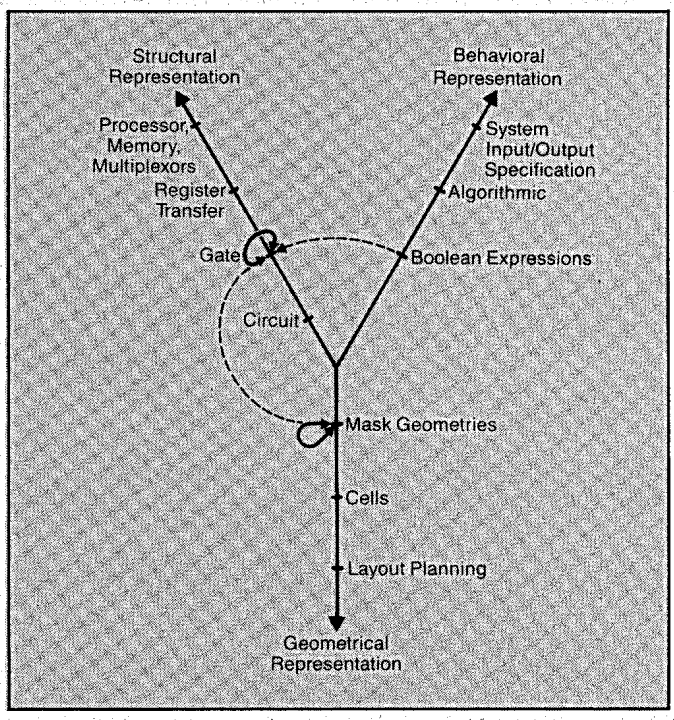


Figure 4: The dotted arcs illustrate CAD tools such as a silicon compiler that transforms one data representation to another.

Another Player in the Arena

Silicon Design Labs, Inc. is developing several products using a silicon compiler technology. The silicon compiler is based on a new language designed specifically for the description of integrated circuits; it supports both circuit attributes (geometry, connectivity, etc.) and computing. For example, Transistor is a language primitive and routing is handled within the language. Programs that can synthesize full-custom circuits are written in this language and can be linked together hierarchically to build larger blocks.

One of the key parameters in efficient silicon compilation is the lowest level language performing the layout. The quality of the behavioral- or structured-level human interface is strongly influenced by the flexibility of the language. If the silicon-level language can only compile rigid or semi-rigid cells and not build an application-specific circuit, then the silicon compiler may not map the designer's exact requirements into silicon. Hierarchy is used to control and manage complexity of SDL's system.

The first initiation of these principles was the PLEX project, undertaken by SDL founders while with Bell Labs and presented at the ICCAD Conference last September. PLEX was a full-custom core microcomputer that could vary the data width, instruction set and word size, data word size, stack depth, etc., depending on the application it was to perform. It also built a unique instruction decoder based on the instructions actually used in the program.

PLEX was a malleable computer that could be tailored to fit a particular instruction set. It could be changed through the use of variables in the silicon compiler. PLEX was a demonstration of how a highly complex digital component could be translated from functional specification to a hand-quality layout with the system designer doing no work at the silicon level. This capability to create full-custom solutions with only a functional description can most readily be accomplished using a compiler technology that supports hierarchies and full parameterization.

The products SDL is developing rely on the concept of hierarchy and a unique layout language. SDL is developing products for both the IC designer and the systems designer that are both design rule and technology independent.

—Peter D. Rip, Vice-President,
Silicon Design Labs, Basking Ridge, NJ
Write 317

ogy. Much research is currently underway to solve this potentially crippling problem. Silicon compilation experts confess that it will be several more years before the size of a silicon compiled die is equivalent to handcrafted designs.

Die size will be the most significant tradeoff when deciding between a silicon compiled or full-custom approach. OEMs making the choice must decide whether small die size is more important than short time to market. Turn-around times for silicon compiled chips should range from a few weeks to a few months. Judging from the present, shorter time to market has the most influence over whether or not a full-custom chip can be implemented into a system.

Laying out a chip is the second area where silicon compilers fall short. Unlike gate arrays whose cells have fixed dimensions or a standard cell which has a fixed height, functional blocks synthesized by most silicon compilers are of variable height and width. The automatic place-and-route algorithms now used to lay out chips are unable to effectively cal-

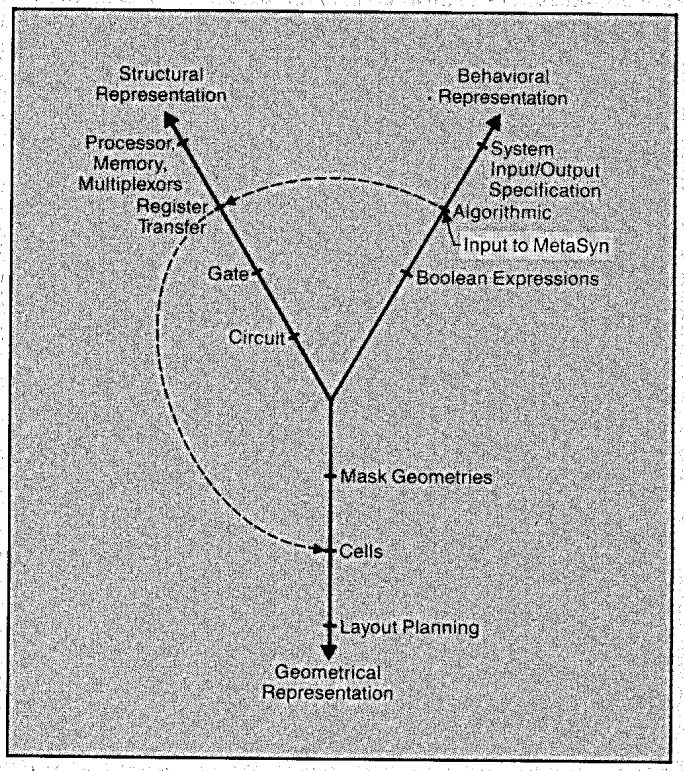


Figure 5: MetaSyn, Metalogic's silicon compiler, accepts design data in the form of an algorithm and transforms it into cell level data.

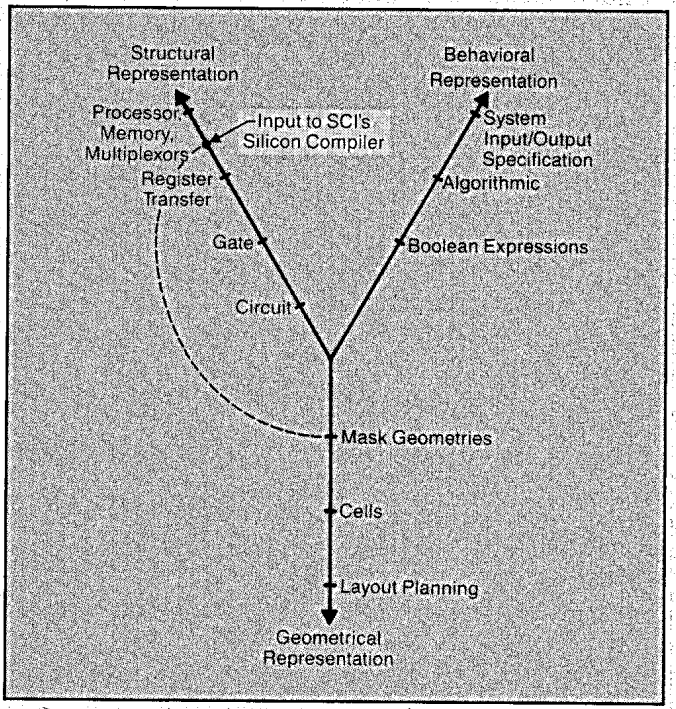


Figure 6: Silicon Compilers Inc. is scheduled to introduce a turnkey silicon compiler next month that will accept structural representations of design data. The compiler will automatically transform this input into mask level data.

The availability of silicon compilation will create a revolution in VLSI design.

culate the most efficient placement of variable sized blocks. In addition, as the number of blocks increases, the time the computer needs to make layout calculations increases exponentially. As a result of this problem, current silicon compilers leave most of the layout task to humans.

Conclusion

The availability of silicon compilation will create a revolution in VLSI design. In essence, silicon compilation is the realization of "computers designing computers." This evolving technology will be the cure for the tremendous demand of custom chips.

Present CAD/CAE design tools such as workstations are bandaids that temporarily ease the VLSI design problem. These tools are quite useful for moderately sized design efforts: the task of designing a chip with 200,000 gates is overwhelming if each gate, flip-flop or register must be drawn. Essentially, schematic capture tools are electronic upgrades of the drafting boards of a manual design methodology.

The real solution to the increasing demand for custom ICs requires tools that remove system architects from logic design. New formulas for VLSI design must be truly top-down, to allow engineers to exploit their design expertise.

References

1) "New VLSI Tools," D. Gajski and R. Kuhn. *IEEE Computer*, Vol. 16, No. 12, December 1983, pp. 11-14.

2) "Automatic Data Path Synthesis," D. Thomas, C. Hitchcock, et al. *IEEE Computer*, Vol. 16, No. 12, December 1983, pp. 59-70. □

How useful did you find this article? Please write in the appropriate number on the Reader Inquiry Card.

Very Useful	621
Useful	622
Somewhat Useful	623

TELL US YOUR THOUGHTS

***Digital Design* is your forum—your inputs help keep the magazine interesting and vital to the design community.** So let us know how we're doing and how we can serve you better in the future. We want to know what you like or dislike about *Digital Design*, the subjects you'd like to see us address, how you feel about the problems you face every day as design professionals.

If you have thoughts your peers should know about, put them in a letter in *Digital Design*. Have your say in *your* magazine! Send letters and comments to: Editor, *Digital Design*, 1050 Commonwealth Ave., Boston, MA 02215.