# HIGH-LEVEL SYNTHESIS OF DIGITAL CIRCUITS

*GIOVANNI DE MICHELI, Stanford University*

VLSI circuits are ubiquitous in electronic design, and they are critical to the progress of the electronic industry. As the market becomes more competitive, designers are being asked to create circuits with increasingly higher performances, zero defects, and high yield, while keeping design and fabrication times to a minimum. Computer-aided synthesis techniques for VLSI circuits have been instrumental in achieving these goals. Commercial implementations of synthesis systems have become practical for the production-level design of digital circuits. In particular, circuit synthesis and optimization techniques have been effectively used to synthesize systems or components from high-level specifications, without or with limited human intervention.

Traditionally, synthesis techniques have been classified in three categories. *Physical design* methods deal with the geometric view of a chip. *Logic synthesis* techniques are used to optimize gate-level representations. Finally, *high-level synthesis* methods generate logic representations from behavioral models. The combination of high-level, logic-level, and physical-level synthesis systems into a hardware compiler—also called a silicon compiler or vertical synthesis system—is one of the major challenges in the design automation field.[1,2]

Of these techniques, tools for physical design are the most common. We understand the problems corresponding to physical design, even though they remain difficult from a computational standpoint.[3] Several design systems are commercially available from vendors or distributed from universities to support physical design. Research in this area is focusing more on the problems resulting from emerging design methodologies, for example, electrically programmable gate arrays.

Logic synthesis techniques have recently matured to the point that designers commonly use systems based on such techniques to analyze and optimize production-level digital designs. In general, logic synthesis systems provide two capabilities: technology-independent circuit optimization and technology mapping into library parts.[4] Most available systems have been conceived for combinational logic design and then been extended to support synchronous circuit design. Sequential logic synthesis techniques for both synchronous and asynchronous circuits are the object of extensive investigation. At present, a common way of entering a digital design is to use programs that capture the logic schematic. The designer's major task then becomes one of transforming the hardware specification into a logic schematic, which is generally validated by simulation.

A problem arises, however, when we try to conceive and enter the logic schematics of large-scale systems. This task is time-consuming, tedious, and error-prone. One way to make it easier is to enter models in an RTL (register-transfer-level) language with structured programming constructs. Even then, the complexity of describing large designs is high and can be reduced only if we raise the abstraction from the logic level to the functional domain.

For these reasons, researchers are concentrating on the third class of synthesis techniques: high-level synthesis. High-level synthesis systems transform a behavioral model of a digital system, written in a hardware description language, to a logic specification. The system provides the designer with either automatic or computer-assisted synthesis. The most widely used HDLs are based on procedural semantics. The transformation of behavioral models into logic-level specification is complex. Designers face—and hence any automatic system faces—many choices in partitioning the system into serial or parallel computation blocks. Partitioning must be done under constraints involving silicon area and timing,[5] which complicates the process.

This special issue is devoted to the recent advances in high-level synthesis. The first article, by R. Camposano, is a tutorial that describes the general principles of high-level synthesis as a stepwise refinement of an original behavioral model. The author shows how we can formalize synthesis from behavioral models as a translation from the modeling language into a graph-based intermediate form. We can then formalize optimization techniques and solve them as discrete graph-optimization problems.

J. Bhasker et al. describe one application of this high-level optimization approach. Algorithmic transformations for high-level optimization are used to search for an implementation that has the maximum performance with the fewest resources. This method uses the intermediate graph model to exploit techniques adapted from software compilers.


0740-7475/90/0010-0001.00©1990 IEEE

The third article gives an example of a practical high-level and logic-level synthesis system, called Olympus. The system supports constraint-driven synthesis, in which timing and resource constraints are applied to guide synthesis decisions. A synthesis-oriented modeling language, called HardwareC, is used to specify behavioral models. Although Olympus is based on algorithmic transformations on an internal design representation similar to those described in the other two articles, it differs from other systems in the synthesis algorithms and in its support of synthesis from behavioral models to netlists of gates in a specified library. In addition, logic synthesis techniques are uniformly incorporated within the high-level synthesis framework to provide estimates to guide high-level decisions.

At present, high-level synthesis techniques are used mostly in research synthesis systems with very few industrial and commercial applications. The future looks promising for this technology, however. I expect high-level synthesis techniques to mature rapidly—just as logic synthesis has done in the last decade—and to be widely used. Many high-level synthesis problems have been well characterized and solved. Nevertheless, the acceptance of high-level synthesis tools for production-level design has been delayed for a number of reasons. First, synthesis is a complex process, and the available tools are still in their infancy. Second, we have not standardized on a synthesis-oriented HDL that can serve as a common starting point for hardware synthesis. True, VHDL (VHSIC hardware description language) has emerged as a standard for hardware specification and simulation, but it has disadvantages. It is a complex language and some of its constructs lack a full definition of hardware semantics. Consequently, we are seeing synthesis approaches from behavioral models in VHDL subsets, or dialects, which defeats the idea of a standard language.

The last—but by no means the least important—reason for the lack of commercial high-level synthesis systems is the acceptance of hardware description in terms of behavioral models. Designers may view such descriptions as unnatural, since they are used to specifying hardware design in all its detail. The introduction of high-level synthesis techniques in hardware design is a lot like the introduction of high-level languages such as C and Fortran in software. Digital designers have to change their approach to design just as programmers did when they abandoned assembly code. To facilitate such a transition, HDLs that support constraints, such as partial structures, have proved to be valid solutions.

Despite these difficulties, I see high-level synthesis systems emerging as a key technology in digital design. Their major advantage is the freedom to explore architectural trade-offs. The automated (or computer-assisted) search for a balanced architecture that matches the implementation technology is extremely important, because the performance and area of a hardware implementation depends much more on high-level optimization than on optimization at the logic and physical design levels. In addition, synthesis techniques from behavioral models to layout will allow designers to integrate larger and larger hardware systems, thus greatly benefitting all microelectronic design.

## REFERENCES


1. *Design Systems for VLSI Circuits: Logic Synthesis and Silicon Compilation*, G. De Micheli, A.Sangiovanni-Vincentelli, and P. Antognetti, eds., Martinus Nijhoff, Dordrecht, The Netherlands, 1987.
2. D. Gajski, *Silicon Compilation*, Addison Wesley, Reading Mass., 1988.
3. E. Kuh and T. Ohtsuki, "Recent Advances in VLSI Layout," *Proc. IEEE*, Vol. 78, No. 2, Feb. 1990, pp. 237-263.
4. R. Brayton, A. Sangiovanni-Vincentelli, and G. Hachtel, "Multi-Level Logic Synthesis," *Proc. IEEE*, Vol. 78, No. 2, Feb. 1990, pp. 264-300.
5. M. McFarland, A. Parker, and R. Camposano, "The High-Level Synthesis of Digital Systems," *Proc. IEEE*, Vol. 78, No. 2, Feb. 1990, pp. 301-318.



**Giovanni De Micheli** is associate professor of electrical engineering—and by courtesy, of computer science—at Stanford University. His research interests include several aspects of CAD for ICs, particularly the automated synthesis, optimization, and verification of VLSI circuits. Previously, he worked at the IBM T.J. Watson Research Center, where he was project leader of the Design Automation Workstation Group. He has also held positions at the Department of Electronics of the Politecnico di Milano, Italy, and at Harris Semiconductor in Melbourne, Florida. He was codirector of the Advanced Study Institute on Logic Synthesis and Silicon Compilation, held in L'Aquila, Italy, under the sponsorship of NATO in 1986 and in 1987.

De Micheli holds a DrEng, summa cum laude, in nuclear engineering from the Politecnico di Milano and an MS and a PhD in electrical engineering and computer science from the University of California, Berkeley. He was granted a Presidential Young Investigator award in 1988 and received the 1987 award for the best paper published in *IEEE Transactions on Computer-Aided Design* and a Best Paper Award from the 1983 Design Automation Conference. He is a senior member of the IEEE and serves as *IEEE D&T*'s editor for synthesis and verification. He was technical chairman of the 1988 International Conference on Computer Design and general chair of the same conference in 1989. His address is Ctr. for Integrated Systems, Stanford University, Stanford, CA 94305.