MICHAEL N. HUHNS

IEEE INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN

ICCAD-86

A CONFERENCE FOR THE EE CAD PROFESSIONAL

NOVEMBER 11–13, 1986
CONVENTION CENTER
SANTA CLARA, CALIFORNIA

IEEE

DIGEST OF TECHNICAL PAPERS

Sponsored by
IEEE COMPUTER SOCIETY
IEEE CIRCUITS AND SYSTEMS SOCIETY
In Cooperation with
IEEE ELECTRON DEVICES SOCIETY
ACM SPECIAL INTEREST GROUP
ON DESIGN AUTOMATION

IEEE Computer Society Order Number 744
Library of Congress Number 86-81578
IEEE Catalog Number 86CH2353-1
ISBN 0-8186-0744-0

Association for Computing Machinery
THE COMPUTER SOCIETY OF THE IEEE
IEEE THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, INC.
IEEE COMPUTER SOCIETY PRESS

Published by IEEE Computer Society Press
1730 Massachusetts Avenue, N.W.
Washington, D.C. 20036-1903

IEEE Computer Society Order Number 744
Library of Congress Number 86-81578
IEEE Catalog Number 86CH2353-1
ISBN 0-8186-0744-0 (paper)
ISBN 0-8186-4744-2 (microfiche)
ISBN 0-8186-8744-4 (case)

Order from: IEEE Computer Society
Post Office Box 80452
Worldway Postal Center
Los Angeles, CA 90080

IEEE Service Center
445 Hoes Lane
Piscataway, NJ 08854

IEEE THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, INC.

Analogical Reasoning for Digital System Synthesis

Ramón D. Acosta, Michael N. Huhns, and Shiuh-li Liuh

Microelectronics and Computer Technology Corporation
9430 Research Blvd.
Austin, Texas 78759

Abstract

Knowledge-based expert systems are being integrated into a variety of VLSI computer-aided design tools. Unfortunately, the static and predetermined capabilities of most of these systems do not allow them to acquire design experience for future use. To overcome this limitation, a digital synthesis system based on analogical reasoning principles has been developed. It is capable of remembering past design experience and employing this accumulated experience in solving new problems. This approach results in a system that acts as an intelligent design assistant because of its ability to improve in both design capabilities and performance through its use.

The system refines VHDL behavioral specifications into structural modules by building hierarchical design trees. This refinement is accomplished by executing plans composed of partially ordered rule sets. The current system has rules for designing circuits comprised of elementary digital components including transistors, logic gates, and inverter loop memory cells. Some of the reasoning and learning techniques being employed include formulation of design plans and their preconditions, abstraction of plans, and transformation of plans to analogous design problems.

1. Introduction

In recent years, several knowledge-based expert systems have been proposed and implemented for aiding in the design of integrated circuits. Some of the problems being investigated include layout [8], simulation [19], verification and analysis [7,9], and synthesis [1,10,15,16,18]. These systems are particularly suited to situations in which heuristic expert knowledge must be employed because algorithmic techniques are unavailable or prohibitively expensive. Unfortunately, the knowledge embodied in these systems is static: it fails to capture the iterative aspects of the design process that involve solving new problems by building upon the experience of previous design efforts.

There have been several attempts to overcome this limitation by making use of more sophisticated reasoning and learning techniques [11,12]. One such system, the Learning Apprentice System (LEAP) for VLSI design [14], acquires knowledge by generalizing from training examples. The transfer of experience from previous design efforts to new problems has also been accomplished via analogical reasoning methods [2,21]. These methods, however, are limited by their requirements that new design problems be identical to previously solved ones.

This paper addresses an analogical reasoning approach to digital design in which old design efforts can aid in solving new problems, even when there is only a partial match between old and new problems. With this approach, a design system can learn either from a user or as a result of its own problem-solving efforts. These ideas have been integrated into a system that refines behavioral descriptions written in VHDL (VHSIC Hardware Description Language) [6,20] to synthesize digital circuits.

2. Design Hierarchy

A plan consists of a sequence of rules that transforms and decomposes a digital circuit design problem by building a hierarchical design tree. Each node of this tree is an entity, or component, that is described in terms of its

(I) Interface – port and parameter declarations,

(B) Behavior – functional specification, and

(S) Structure – subcomponents and their interconnections.

The description of an entity's interface and behavior (I-B) specify a design problem. A rule can be applied when its antecedent matches an I-B. Executing its consequent results in building a structural description (S) for the I-B, as well as all of the subcomponent descriptions (I-Bs) instantiated by S. A sequence of rule applications refines this specification into a tree of I-B-S nodes. The leaves of this tree constitute a solution synthesized in terms of instantiated library components, such as standard cells or parameterized modules. Fig. 2 shows an I-B-S tree constructed during the design of the simple circuit in Fig. 1.

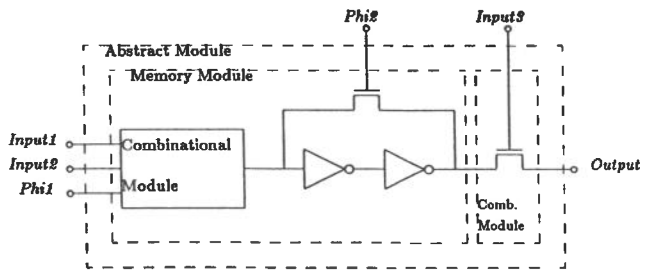

Fig. 1. Schematic diagram for a circuit comprised of a memory element and combinational logic.

CH2353-1/86/0000/0173$01.00 © 1986 IEEE

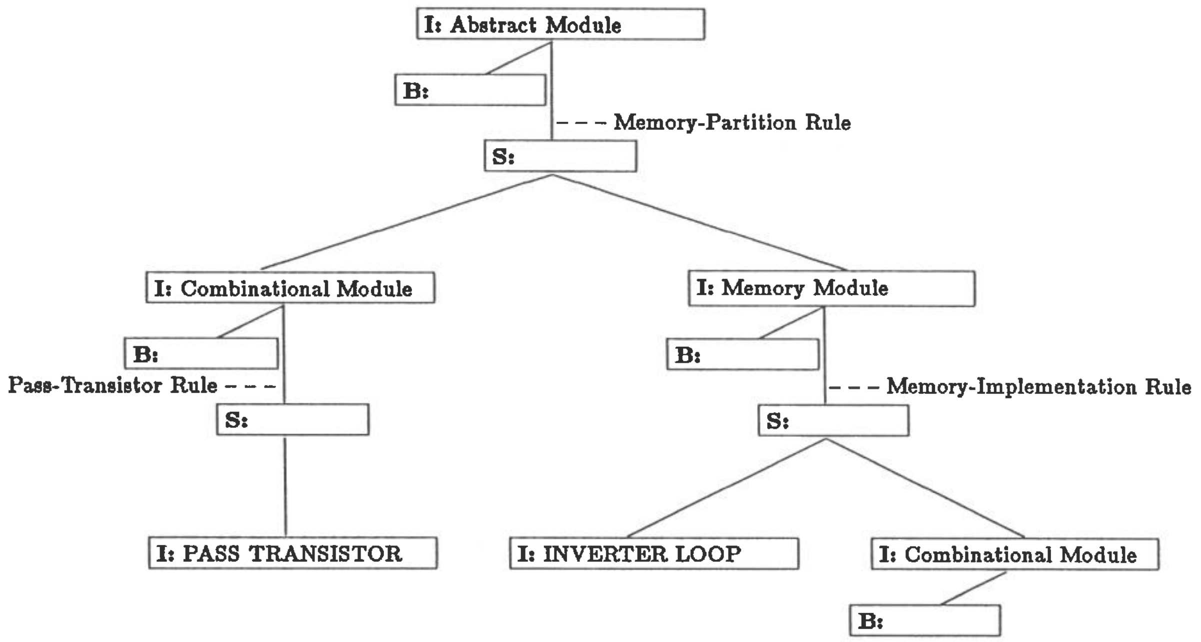

Fig. 2. Hierarchical design tree for the circuit in Fig. 1.

The system uses VHDL for representing VLSI designs. Since the declarative facilities of VHDL are intended for describing design abstraction, they are well suited for specifying the I-B-S design hierarchy.

3. Strategy for Solving Design Problems

A variety of artificial intelligence techniques is available for solving circuit design problems. The following control strategy for solving a problem *P* is based on classifying these techniques according to the amount and specificity of domain knowledge they require [2]:

(1) If knowledge of a circuit that satisfies *P* is available, then this solution is directly instantiated.

(2) If a specific plan for transforming and decomposing *P* is available, then it is directly executed.

(3) If a plan for solving *P′* is available, where *P′* is "similar" to *P*, then it is analogically transformed to synthesize a circuit for *P*.

(4) If past experience is unavailable, then weak methods such as heuristic search or means-ends analysis are employed.

To implement this strategy, a system must be capable of formulating, remembering, and executing design plans.

4. Formulating Plans and Plan Preconditions

In the system described herein, circuit design rules are applied deductively to transform and decompose VHDL specifications of design problems. Because a hierarchy of independent subentities is created by this process, rule sets leading to final designs are partially ordered when they are saved as plans (Fig. 3). These plans can be followed to design circuits for new problem specifications.

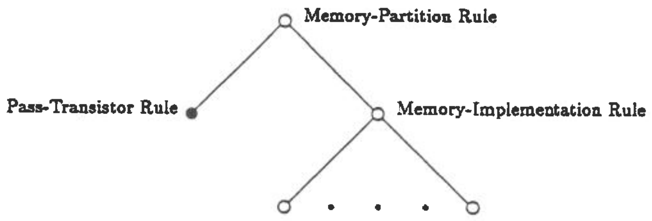

Fig. 3. Plan for synthesizing the circuit in Fig. 1.

The precondition for a plan is a set of conditions that a problem's specification must meet in order for the plan to be applicable. Previous techniques, such as the STRIPS triangle table [3,4] or goal regression in explanation-based generalization [13], derive these conditions based on a plan and specific training examples. The approach used here differs in that only the partially ordered rules of the plan are used. Thus, the most general (i.e., necessary and sufficient) precondition is obtained.

5. Reasoning by Analogy

Exact analogies arise when the specification of a new problem satisfies the precondition for a plan of a previously solved design problem. Yet it is conceivable that an old plan might almost apply, except for a few minor steps. In order to make use of the previous problem-solving experience embedded in a plan, it should be possible to use portions of it to solve, at least partially, the new problem. Thus, the use of inexact analogies via increasingly abstract design plans has been developed.

Plan abstractions [5,17] are obtained by omitting or generalizing steps that are considered details (Fig. 4). Consequently, abstracted plans have more general preconditions and are applicable to a wider class of problems. These

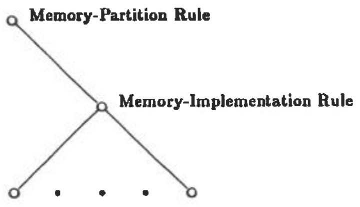

Fig. 4. Abstraction for the plan in Fig. 3.

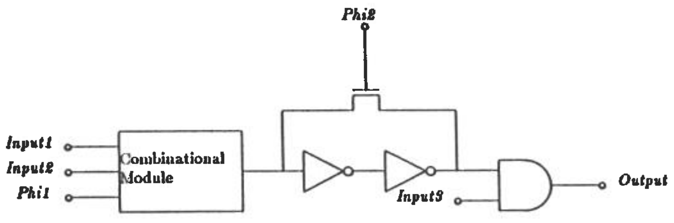

Fig. 5. Analogous circuit to that in Fig. 1.

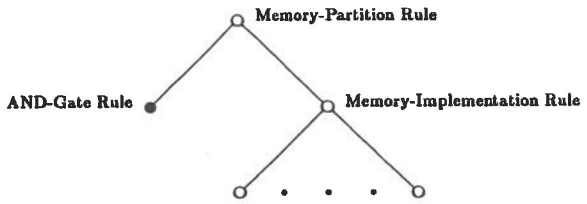

Fig. 6. Analogical transformation of the abstract plan in Fig. 4.

plans are analogically transformed to suit specific problems by appropriately filling in missing steps (Figs. 5-6). Note that as plans get more abstract, the number of details that must be filled in to solve a given problem increases. Hence, it is desirable to find and apply the least abstract plan that is appropriate.

Three activities can be identified in this abstraction procedure:

(1) generating and storing plan and precondition abstractions from given design problems and solutions,

(2) recognizing analogies by matching new problem specifications with abstract preconditions for previously solved problems, and

(3) analogically transforming abstract plans to synthesize solutions to new problems.

Because of the inherent complexity of VLSI design, interactive user assistance for various phases of this process is employed. With use, the system accumulates design knowledge resulting in an increase in both its performance and its ability to synthesize digital circuits.

1. Results

The current system employs rules written in Common Lisp. Table 1 shows some measurements of execution time and rule use for a small test circuit. Three strategies were evaluated on a Symbolics 3640:

(S1) exhaustively computing all possible design plans,

(S2) exhaustively computing all possible specializations for an abstract plan, and

(S3) directly applying a specific plan.

The abstract plan, which consisted of three rules, was obtained by eliminating the rules that instantiated library components in the nine-rule specific plan used in strategy (S3). Designing with the aid of an abstract plan resulted in reductions of over 98% in execution time, rules applied, and number of alternative designs produced. While these results do not reflect searching for previous plans or using heuristics to reduce the exhaustive search, they strongly suggest that dramatic improvements in the performance of design systems can be achieved by incorporating analogical reasoning techniques.

Table 1. Test Circuit Measurements (CAM cell)

Design Strategy	Execution Time (sec)	Rules Attempted	Rules Applied	Number of Designs
(S1)	61232.7	7145025	117570	50463
(S2)	770.3	57753	1519	616
(S3)	5.3	9	9	1

1. Conclusions

A system has been developed for the design of digital systems through the refinement of behavioral specifications into structural specifications using analogical reasoning. Analogical transformation of increasingly abstract design plans is used when more specific plans fail to apply. The main advantages of this system are that it improves in both performance and design capabilities as it is used.

At present, the system has rules for designing circuits comprised of elementary digital components including transistors, logic gates, and clocked inverter loop memory cells. It is being extended to include rules for the synthesis of processor-level circuits using higher-level building blocks.

3. References

1. H. Brown, C. Tong, and G. Foyster, "Palladio: An Exploratory Environment for Circuit Design," *Computer*, Vol. 16, No. 12, December 1983, pp. 41-56.

2. J.G. Carbonell, "Derivational Analogy: A Theory of Reconstructive Problem Solving and Expertise Acquisition," in *Machine Learning: An Artificial Intelligence Approach, Vol. II*, R.S. Michalski, J.G. Carbonell, and T.M. Mitchell, Eds., Morgan Kaufmann, Los Altos, CA, 1986, pp. 371-392.

3. R.E. Fikes, P. Hart, and N.J. Nilsson, "Learning and Executing Generalized Robot Plans," *Artificial Intelligence*, Vol. 3, 1972, pp. 251-288.

4. R.E. Fikes and N.J. Nilsson, "STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving," *Artificial Intelligence*, Vol. 2, 1971, pp. 189-208.

5. P.E. Friedland and Y. Iwasaki, "The Concept and Implementation of Skeletal Plans," *Journal of Automated Reasoning*, Vol. 1, 1985, pp. 161-208.

6. *IEEE Design & Test of Computers*, Special Issue on VHDL: The VHSIC Hardware Description Language, Vol. 3, No. 2, April 1986.

7. V.E. Kelly, "The CRITTER System - Automated Critiquing of Digital Circuit Designs," *Proceedings of the 21st Design Automation Conference*, June 1984, pp. 419-425.

8. J. Kim and J. McDermott, "Computer Aids for IC Design," *IEEE Software*, Vol. 3, No. 2, March 1986, pp. 38-47.

9. A. Kolodny, R. Friedman, and T. Ben-Tzur, "Rule-Based Static Debugger and Simulation Compiler for VLSI Schematics," *Proceedings of the IEEE International Conference on Computer-Aided Design*, 1985, pp. 150-152.

10. T.J. Kowalski, *An Artificial Intelligence Approach to VLSI Design*, Kluwer Academic Publishers, Boston, Massachusetts, 1985.

11. R.S. Michalski, J.G. Carbonell, and T.M. Mitchell, Eds., *Machine Learning: An Artificial Intelligence Approach, Vol. I*, Tioga, Palo Alto, California, 1983.

12. R.S. Michalski, J.G. Carbonell, and T.M. Mitchell, Eds., *Machine Learning: An Artificial Intelligence Approach, Vol. II*, Morgan Kaufmann, Los Altos, California, 1986.

13. T.M. Mitchell, R.M. Keller, and S.T. Kedar-Cabelli, "Explanation-Based Generalization - A Unifying View," Technical Report ML-TR-2, Laboratory for Computer Science Research, Rutgers University, August 1985.

14. T.M. Mitchell, S. Mahadevan, and L.I. Steinberg, "LEAP: A Learning Apprentice for VLSI Design," *Proceedings of the 9th International Joint Conference on Artificial Intelligence*, August 1985, pp. 573-580.

15. T.M. Mitchell, L.I. Steinberg, and J.S. Shulman, "A Knowledge-Based Approach to Design," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, Vol. PAMI-7, No. 5, September 1985, pp. 502-510.

16. P.S. Rosenbloom, J.E. Laird, J. McDermott, A. Newell, and E. Orciuch, "R1-Soar: An Experiment in Knowledge-Intensive Programming in a Problem-Solving Architecture," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-7, No. 5, September 1985, pp. 561-569.

17. E.D. Sacerdoti, "Planning in an Hierarchy of Abstraction Spaces," *Artificial Intelligence*, Vol. 5, No. 2, 1974, pp. 115-135.

18. E. Simoudis and S. Fickas, "The Application of Knowledge-Based Design Techniques to Circuit Design," *Proceedings of the IEEE International Conference on Computer-Aided Design*, 1985, pp. 213-215.

19. N. Singh, "MARS: A Multiple Abstraction Rule-Based Simulator," Memo No. HPP-83-43, Stanford Heuristic Programming Project, December 1983.

20. "VHDL Language Reference Manual, Version 7.2," Intermetrics Report IR-MD-045-2, August 1985.

21. P.H. Winston, "Learning and Reasoning by Analogy," *Communications of the ACM*, Vol. 23, No. 12, December 1980, pp. 689-703.