Carnegie Mellon University
Research Showcase @ CMU

Department of Electrical and Computer Engineering

Carnegie Institute of Technology

1982

The VSLI design automation assistant : first step

Thaddeus J. Kowalski
Carnegie Mellon University

Donald E. Thomas

Follow this and additional works at: http://repository.cmu.edu/ece

Recommended Citation
.

This Technical Report is brought to you for free and open access by the Carnegie Institute of Technology at Research Showcase @ CMU. It has been accepted for inclusion in Department of Electrical and Computer Engineering by an authorized administrator of Research Showcase @ CMU. For more information, please contact research-showcase@andrew.cmu.edu.

NOTICE WARNING CONCERNING COPYRIGHT RESTRICTIONS:
The copyright law of the United States (title 17, U.S. Code) governs the making of photocopies or other reproductions of copyrighted material. Any copying of this document without permission of its author may be prohibited by law.

THE VLSI DESIGN AUTOMATION ASSISTANT: FIRST STEP

by

T.J. Kowalski i D.E, Thomas

Doeembjr, 1932

DRC-18-57-32

UNIVERSITY LIBRARIES V
CARNEGIE-MELLON UNIVERSITY
PI[illegible] 15213

The VLSI Design Automation Assistant: First Steps

T. J. Kowalski
D. E. Thomas

Electrical Engineering Department
Carnegie-Mellon University

ABSTRACT

This paper describes an approach to VLSI design synthesis using both knowledge-based expert systems and data and control flow analysis. We are concerned with design synthesis as it proceeds from an algorithmic description of a VLSI system to a list of technology-independent registers, operators, data paths, and control signals. This paper discusses the development of the Design Automation Assistant from its first interviews with expert VLSI designers to its current prototype state. Four designs of a microcomputer are presented along with the changes in the knowledge base that created those designs.

INTRODUCTION

Recent advances in integrated circuit fabrication technology have allowed larger and more complex designs to form a complete system[1] on a single VLSI chip. These chips use one to five micron features to achieve complexities with an equivalence of one hundred thousand to two hundred fifty thousand transistors. This level of design complexity has created a combinatorial explosion of details, which is a major limitation in realizing cost-effective, low-volume, special-purpose VLSI systems. To overcome this limitation, tools and design methods capable of exploiting the hierarchical and constraint properties of design must be built.

We have been developing just such synthesis tools[2] for aiding the designer in developing the algorithmic description of the hardware and interactively adding the details required to produce a finished design. This structured approach can decrease the time it takes to design a chip, automatically provide multi-level documentation for the finished design, and create reliable and testable designs.

This paper focuses on the synthesis, or allocation, of the architectural-design space. It proceeds from an algorithmic description of a VLSI system to a list of technology-independent registers, operators, data paths and control signals. Because of the combinatorial explosion of details and implicit-dynamic constraints used to choose an architecture, this problem does not lend itself to a recipe-like solution. However, Knowledge-Based Expert Systems[3] (KBES) provide a framework for just such problems that can be solved only by experts using large amounts of domain-specific knowledge to focus on specific design details and constraints. This paper discusses the development of the Design Automation Assistant (DAA) from its conception to a snap-shot of the current system. The development is discussed in three parts: gathering "book knowledge", codifying the knowledge into a prototype knowledge-base system, and refining the knowledge base. Four designs of a MOS Technology Incorporated MCS6502 microcomputer are presented along with the changes in the knowledge base that created those designs.

1. CONCEPTION

Development of most KBES's have been attempted in several stages. First, "book knowledge" of the problem is codified as a set of situation-action rules using interviews with experts to fill in missing knowledge and refine already existing knowledge. Then a large volume of example problems are given to the KBES and the results are closely examined by experts to validate the results. Often, errors are found by the examples and new rules are added to the system to handle the situations. This iterative process is necessary because experts are often unaware of exactly how they go about designing a chip and inexperienced at articulating the steps they go through. Also, the knowledge base is not an exact codification of the expert's knowledge, but what is understood by the knowledge engineer.

After gathering the current "book knowledge*[9] about synthesis of the architectural design space,[4*5,6] we interviewed four designers of varied experience: one was a novice, two were moderately experienced and one was an expert. The interviews, which lasted about an hour each, started with a discussion of the designer's background including: years of experience, logic families used, and designs done. Most of the time was spent discussing the design process with some time given to a discussion of the DAA system. We used an interview method that allowed the interviewees as much freedom as possible in generating ideas by placing a strong emphasis on questions like:

"What do you do next?[91]
and
"Could you elaborate on ... ?"

The designers discussed the global picture, partitioning, selection, and allocation tasks. They began with a high-level overview of the hardware's function, which listed inputs and outputs to the outside world, functions the hardware should provide, general constraints, and design feasibility with consideration of the target technology. They generally partitioned the global picture into smaller blocks with emphasis on minimizing connections between blocks, selecting blocks that operated as parallel or serial units, and selecting groups that have similar function. Partitions were chosen for allocation in a decreasing order of difficulty, or amount of constraint. The designers reasoned that if the most difficult part could be designed, the rest of the design was possible.

Once a partition was selected for allocation, it was either carried out in parallel, or in serial. A parallel design made thinking of the control logic much simpler, while a serial design would minimize design area. The constraints of the parallel design were examined for size violations to determine what parts should be serialized by adding data paths, registers, and control logic to the initial parallel design; the constraints of the serial design were examined for speed violations to determine what parts should be reimplemented in parallel. If they recognized a part of the design that was similar to a previous design, they used what they knew had worked in the past. Within each partition designers allocated clock phases, operators, registers, data paths, and control logic. The order was interesting because once registers and data paths were allocated they were not changed. The control was changed because it was the hardest to think about and it depended on a constant structure for the data-path elements.

The designers described the iteration process of a design as a step-by-step refinement to meet violated constraints. The designers looked for a technology change to meet a constraint before making a change to their design. This could be as simple as finding a new chip in the TTL data book, or as complicated as a design rule shrink. Next they would give up functionality to meet a constraint. One designer summed it up best by saying,

> "An engineer's training teaches him when constraints can be swept under the rug.[99]

The relative importance of constraints are application dependent. Constraints the designers mentioned were: speed, area, power, schedule, cost, drive capabilities, and bit width. Lastly, other design changes consisted of global improvements that were not recognized until the design neared completion. This suggests that the general choice of partitions and the initial design style selections were close to optimum.

2. BIRTH

Even though there were many details missing, we had gathered enough [t4]book knowledge[99] to put together a prototype version of the DAA system using the OPS5 KBES writing system.[7] The initial knowledge in the system was codified from the algorithms of the current CMU/DA allocator[4] and the interviews discussed above. While the prototype system served as a stimulus for further elicitation sessions with expert designers, it was far from perfect. This section discusses the flow of control in the prototype system.

DAA begins by allocating the base-variable storage elements to hardware modules and ports; base-variable storage elements are constants, architectural and global registers, and memories with their input, output and address registers. Then a data-flow BEGIN/END block is picked and the synthesis operation assigns data-flow operators to clock phases using minimum delay information to develop a parallel design. Next,

it assigns temporary registers to all data-flow operator outputs not bound to base-variable storage elements. Lastly, it assigns temporary modules, ports, and links to each data-flow operator, avoiding multiple assignments of hardware links and supplying multiplexers where necessary.

After a data-flow BEGIN/END block has been synthesized, the hardware modules are a worst-case allocation of temporary modules and registers. An analysis operation starts and removes temporary registers from the data-flow outputs where the sources of the data-flow operator are stable. It also combines temporary modules of the same type and size within the synthesized block. Finally, any remaining temporary module or register is made permanent and the process is repeated for the next BEGIN/END block.

3. FIRST STEPS

The first prototype DAA system had about 70 rules and could design a MOS Technology Incorporated MCS6502 microcomputer in about 3 hours of VAX 11/750 CPU time. We asked many expert designers at INTEL and Bell Laboratories to critique the design by explaining what was wrong, why it was wrong, and how to fix it. After each critique rules were modified, new rules were added, and the MCS6502 was re-designed. As of the writing of this draft, DAA had 130 rules and designed a much better MCS6502 microcomputer in about 4 hours of VAX 11/750 CPU time. In retrospect, much of what we learned was common-sense knowledge, the same knowledge human designers learn through apprenticeship. Though DAA has undergone many improvements and produced many designs of the MCS6502 microcomputer, this section presents four designs that are intended to represent the knowledge taught through many interviews.

Each interview started by giving a designer a drawing of the design with a sheet of clear plastic over it. As the designer started giving the critique, pieces of cardboard were placed over the design, so we could tell what parts of the design the designer was looking at. Whenever the designer made a correction to the design, a new sheet of plastic was put down. The designers found this elidtation procedure agreeable with their normal spatial mode of operation.

TABLE 1. MCS6502 - FOUR DESIGNS

Designs	1	2	3	4
Awl	20	12	20	20
Cop	177	85	**25**	1
Minos	64	9	9	0
Or	9	1	9	9
Not	21	21	21	21
Plus	540	121	77	0
sm	9	0	0	0
Sir	9	0	0	0
SrO	8	0	0	0
Srr	9	0	0	0
Xor	9	1	9	9
Ahi	0	45	81	35
Dree	450	450	450	450
Treg	1227	288	315	292
Max In	2122	2698	2791	2706
Max Out	293	614	524	360

The first design used the prototype DAA design system described in Section 2. Column 1 of Table 1 summarizes certain characteristics of this design. Each row shows the bits of the specified operator or register type found in the design. The experts* critique included:

- It didn't share modules across blocks.
- One-bit modules within the same block should not be combined because MUXes are more expensize than most one-bit modules.
- Registers could increment, decrement and shift their values.
- Temporary registers to the controller should be eliminated and one latched register should be placed in front of the controller.
- The wiring data-flow operators should not be combined; the wiring data-flow operators are CONCAT, PAD0, and PADS.

The rules were changed and partitioning information was added, based on connectivity of data-paths and similarity of operators between blocks, to simplify the combining of modules of the same type between blocks. Column 2 of Table 1 summarizes the changes resulting from the critique. The comparisons (CMP), additions (PLUS), and temporary registers (TREG) declined although the multiplexers increased. The bits of AND, OR, and XOR also decreased. The next set of critiques included:

- Logical data-flow operators should not be combined because MUXes are more

expensize than most logical modules; logical data-flow operators are: AND, OR, and NOT.

- They thought it is less expensive to group modules of different types together to form ALU modules.

Rules were added to help the combination of modules of different sizes and types. As can be seen in column 3, the ALU number increased, further decreasing the PLUS and CMP numbers. The bits of AND, OR, and XOR returned to their original values. The critiques now cited:

- The requirement to control the parallelism/serialism of a design by specifying the instantiations of a certain operator type, and the clocks steps for a complete instruction.
- Also combining all operators together into an ALU for small designs like the MCS6502.

Rules were modified to check for violation of too many modules for each type, thus forcing the changing of clock phase assignments to data-flow operators. Also changes were made to check for the length of the control path exceeding the maximum allowable length. A further decrease is noted in the numbers of operators in column 4. Now the complaints were:

- Temporary registers should be combined when not actively holding a value.
- Some registers that serve to multiplex values into data-flow blocks should be removed, leaving just the multiplexers into the blocks.
- Multiplexers that have more than *N* ports should be modified into bus structures

These and other critiques are currently being included in the system.

4. CONCLUSION

We are exploring the allocation problem of operators, registers, data paths and control paths from an algorithmic representation of a VLSI system by using a KBES to test the knowledge gathered from interviews with experts to create interesting and usable designs. Using expert systems has allowed incremental addition of modular knowledge and queries about that knowledge during the design task. The CMU/DA system provides global constraints and local partitioning information, which allows DAA to synthesize designs that are both globally and locally optimized. This research has added primarily to knowledge in the digital design synthesis domain by enumerating the set of rules used by expert designers. Secondarily, it has added to knowledge in the expert system domain by providing another system for researchers to examine. Understanding the cognitive processes of expert VLSI designers provides better synthesis tools, thus, decreasing the cost of low-volume special-purpose chips and reducing the design time. This knowledge will also aid in the teaching of design by making explicit, knowledge that is now only passed on through apprenticeship. Our future plans are to have DAA design a VAX computer and use experts at Digital Equipment Corporation to critique the design.

ACKNOWLEDGEMENT

We would like to thank K. Chong, D. Ditzel, M. Maul, G. Mowery, A. Ross, C. Schneider, G. Williams, and A. Wilson for donating time to critique the various designs.

REFERENCES

[1] Mead, Carver and Conway, Lynn, *Introduction to VLSI systems,* Addison-Wesley Publishing Company, Reading, Massachussetts (1980).

[21 Director, S. W., Parker, A. C, Siewiorek, D. P., and Thomas, D. E., "A design methodology and computer aids for digital VLSI systems," *IEEE Transactions on Circuits and Systems* cas-28(7)(July, 1981).

[3] Feigenbaum, E. A., *Knowledge Engineering: The Applied Side of Artificial Intelligence,* Computer Science Department, Stanford University (1980).

[4] Lou Hafer, *Data - Memory Allocation in the Distributed Logic Design Style,* Masters thesis, Carnegie-Mellon University (December 21f 1977).

[5] Marwedel, P. and Zimmermann, G., *MIMOLA Software System User Manual,* 1, Institut Fur Informatik und Praktische Mathematik, Christian-

Albrechts-Universitat Kiel (May, 1979).

[6] Hafer, L. J., *Automated data-memory synthesis: A format Method for the Specification, Analysis, and Design of Register-Transfer Level Digital Logic,* PhD thesis, Department of Electrical Engineering, Carnegie-Mellon University (June, 1981). Also in Design Research Center DRC-02-05-81

[71 Forgy, C. L., *OPS5 User's Manual.* Department of Computer Science, Carnegie-Mellon University (July, 1981).