VLSI ELECTRONICS

Microstructure Science

14

Edited by
NORMAN G. EINSPRUCH

VLSI Design

VLSI Electronics Microstructure Science

Volume 14

VLSI Design

Edited by

Norman G. Einspruch

College of Engineering
University of Miami
Coral Gables, Florida

1986

ACADEMIC PRESS, INC.

Harcourt Brace Jovanovich, Publishers

Orlando San Diego New York Austin
Boston London Sydney Tokyo Toronto

ACADEMIC PRESS, INC.
Orlando, Florida 32887

United Kingdom Edition published by
ACADEMIC PRESS INC. (LONDON) LTD.
24–28 Oval Road, London NW1 7DX

Library of Congress Cataloging in Publication Data
Main entry under title:

VLSI design.

(VLSI electronics : microstructure science ; v. 14)
Includes bibliographies and index.
1. Integrated circuits–Very large scale integration –Design and construction. I. Einspruch, Norman G. II. Series: VLSI electronics ; v. 14.
TK7874.V56 vol. 14 621.395 s [621.395] 85-30669
ISBN 0–12–234114–7 (alk. paper)

PRINTED IN THE UNITED STATES OF AMERICA

86 87 88 89 9 8 7 6 5 4 3 2 1

VLSI Electronics Microstructure Science

Volume 14

VLSI Design

List of Contributors

Numbers in parentheses indicate the pages on which the authors' contributions begin.

Antun Domic (115), MIT Lincoln Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139

Michael J. Foster (139), Department of Computer Science, Columbia University, New York, New York 10027

J. W. Gannett* (81), AT&T Bell Laboratories, Murray Hill, New Jersey 07974

John L. Hennessy (1), Computer Systems Laboratory, Department of Electrical Engineering, Stanford University, Stanford, California 94305

Manolis G. H. Katevenis† (35), Computer Science Division, Electrical Engineering and Computer Sciences, University of California, Berkeley, California 94720

David A. Patterson (35), Computer Science Division, Electrical Engineering and Computer Sciences, University of California, Berkeley, California 94720

Steven A. Przybylski (1), Computer Systems Laboratory, Department of Electrical Engineering, Stanford University, Stanford, California 94305

Carlo H. Séquin (35), Computer Science Division, Electrical Engineering and Computer Sciences, University of California, Berkeley, California 94720

Robert W. Sherburne, Jr.‡ (35), Computer Science Division, Electrical Engineering and Computer Sciences, University of California, Berkeley, California 94720

* Present address: Bell Communications Research, Morristown, New Jersey 07960.

† Present address: Computer Systems Laboratory, Computer Science Department, Stanford University, Stanford, California 94305.

‡ Present address: Electrical, Computer and Systems Engineering Department, Rensselaer Polytechnic Institute, Troy, New York, 12181.

Preface

Civilization has passed the threshold of the second industrial revolution. The first industrial revolution, which was based upon the steam engine, enabled man to multiply his physical capabilities to do work. The second industrial revolution, which is based upon semiconductor electronics, is enabling man to multiply his intellectual capabilities. VLSI (Very Large Scale Integration) electronics, the most advanced state of semiconductor electronics, represents a remarkable application of scientific knowledge to the requirements of technology. This treatise is published in recognition of the need for a comprehensive exposition that assesses trends for the future of VLSI electronics and the scientific base that supports its development.

These volumes are addressed to scientists and engineers who wish to become familiar with this rapidly developing field, basic researchers interested in the physics and chemistry of materials and processes, device designers concerned with the fundamental character of and limitations to device performance, systems architects who will be charged with tying VLSI circuits together, and engineers concerned with utilization of VLSI circuits in specific areas of application.

This treatise includes subjects that range from microscopic aspects of materials behavior and device performance — through the technologies that are incorporated in the fabrication of VLSI circuits — to the comprehension of VLSI in systems applications.

The volumes are organized as a coherent series of stand-alone chapters, each prepared by a recognized authority. The chapters are written so that specific topics of interest can be read and digested without regard to chapters that appear elsewhere in the sequence.

There is a general concern that the base of science that underlies integrated circuit technology has been depleted to a considerable extent and is in need of revitalization; this issue is addressed in the National Research Council

(National Academy of Science/National Academy of Engineering) report titled "Microstructure Science, Engineering and Technology." It is hoped that this treatise will provide background and stimulus for further work on the physics and chemistry of structures that have dimensions that lie in the submicrometer domain and the use of these structures in serving the needs of humankind.

Contents

VLSI ELECTRONICS: MICROSTRUCTURE SCIENCE, VOL. 14

Chapter 1

VLSI Processor Design Methodology*

JOHN L. HENNESSY AND STEVEN A. PRZYBYLSKI

Computer Systems Laboratory
Department of Electrical Engineering
Stanford University
Stanford, California

* The MIPS processor design, which is used as an example in this paper, has been supported by the Defense Advanced Research Projects Agency under grants MDA903-79-C-680 and MDA903-83-C-0335.

I. INTRODUCTION

Integrated circuit technology has made possible the production of chips with hundreds of thousands of transistors. Systems of such complexity remain difficult to design. The computer architect faces problems in the areas of system partitioning with subgoal specification, subsystems interface specification and verification, and overall system integration.

This improvement in integrated circuit technology allows the fabrication of processors with complexity comparable to the largest mainframe computers designed using off-the-shelf technologies (SSI, MSI, and LSI). These mainframe machines, such as the Cray-1, the IBM 360/91, and the CDC 7600, have extremely long design cycles and, as a result, high design costs. With a microprocessor, a low selling price combined with a fairly short product life cycle may make such large design expenditures difficult to justify. Additionally, the technology changes so fast that the long design cycle and optimality of the design may be negated by the rapidly improving technology.

The use of VLSI as an implementation medium establishes several ground rules.

(1) Correctness of the design is of paramount importance. Debugging a flawed chip design is both difficult and time consuming. Alterations cannot be immediately tested, but must wait for a period of weeks to months. This forces batching of changes to chip and a high insistence on nearly perfect designs.

(2) The degree of flexibility in the design is incredibly high; the designers specify the system organization, the partitioning, the physical placement, and even the details of the individual driving logic and gate transistors.

(3) Despite this flexibility, there are limitations in the ability of one level of the design to compensate for shortcomings at higher levels. These limitations come from both inherent constraints in the technology (size, power, and speed), as well as the need to limit the addition of new complexity at lower levels of the design. Performance becomes an issue that must be addressed at all levels of the design.

Throughout this chapter, we will concentrate on the design of general-purpose microprocessors. Though the specific trade-offs may vary, the concepts and techniques apply directly to special-purpose VLSI processors as well. Since the MOS technologies have been the primary vehicle for commercial microprocessors we emphasize MOS design methodologies. We will consider the problem of VLSI computer design as four separate, albeit heavily interrelated, tasks: architectural specification, processor organization, physical design, and electrical implementation.

Specification of the architecture involves the definition of the instruction set and the behavior of all components visible to the user of the processor. These invariably include the register file, the functional units, and the memory units — including a specification of addressing modes. The nature of and interface to the exception system is also included and, in some architectures, the input/output interface. The architecture is largely implementation independent: it does not preclude the choice of any particular technology, though it may strongly favor one technology over another.

An implementation of the architecture begins to take shape by defining the processor's logical organization. The interconnection of functional units is linked with a control structure and a per-cycle timing description that specifies the sequence of operations that will implement each activity in the architecture. The details of the memory hierarchy and mapping need to be defined. Portions of this part of the design may be architecturally transparent (such as the existence of caches), while other parts (such as the memory mapping scheme) may be defined by the architecture. Given all this information, the performance of the processor can be calculated with fair accuracy in terms of numbers of clock cycles per instruction.

Physical design is the process of partitioning of the entire CPU and memory system onto the physical units that make up the system: racks, cards, and chips. With the introduction of VLSI, physical decomposition becomes even more significant: the high relative cost of crossing chip boundaries makes the definition of these boundaries crucial. Within a single integrated circuit, functional blocks must be arranged to accommodate the limited area resources and the interconnection constraints of the implementation technology. We also include in the physical design process the task of decomposing the functional blocks into the logic networks that implement them.

By electrical design we mean the translation of logic diagrams to transistor networks and the accompanying task of layout. We shall deal primarily with the use of sophisticated circuit design techniques to attain high performance and the problems of power and area management.

In the past these four aspects of processor design have frequently been dealt with separately. The effects of the implementation technology on the logical and architectural levels were kept to a minimum. Likewise, once a gate level design was specified, most of the higher-level aspects of the design were ignored in the physical and detailed electrical design. The advantage of this linear decomposition was that the design process took on a more top-down, serial nature. The result was to limit the complexity of the overall process to a manageable level. The development group was made up of distinct teams of people working separately on the various tasks, passing their results to those further down the conceptual ladder.

The emergence of MOS VLSI as a viable implementation technology has brought with it a considerable changing and strengthening of the interactions

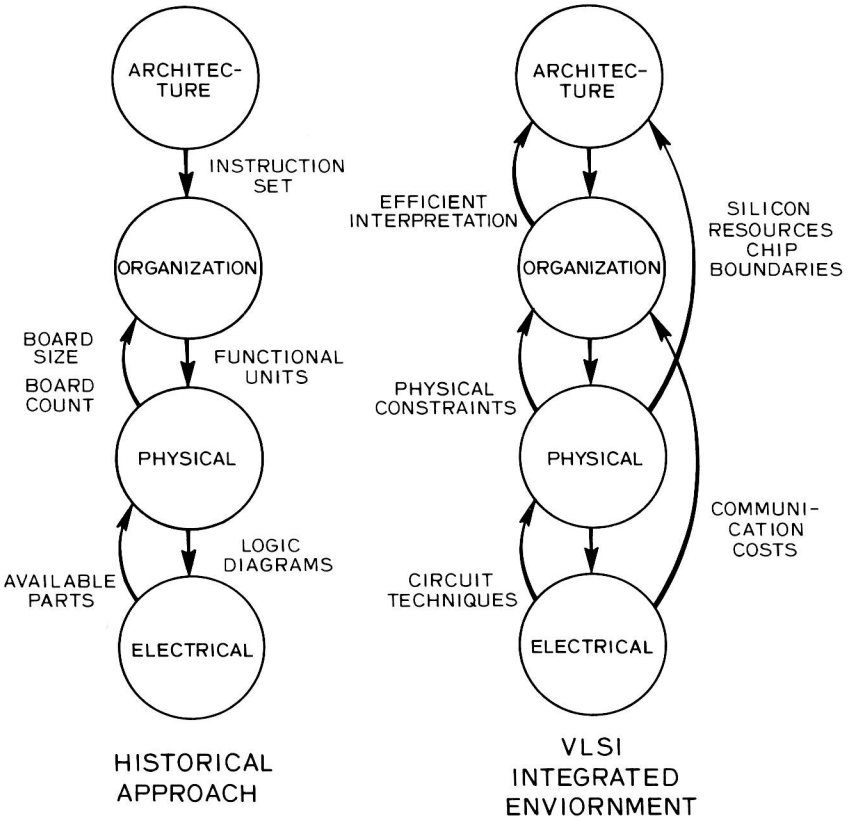

Fig. 1. Decomposition of the design process.

between the various design levels. Figure 1 illustrates some of these constraints and influences. If these additional upward interactions are not properly and consistently addressed, the resulting computer will not attain its full performance potential.

An integrated approach to processor design involves the simultaneous development of:

(1) an architecture,
(2) an organization that efficiently decodes and executes the instruction set architecture and that maps well onto silicon, and
(3) a timing framework that tightly couples the organization with prior knowledge of the circuit techniques to be used along the expected critical paths.

Having the various aspects progress in parallel leads to a final design that makes more efficient use of the implementation technology and will thus outperform more naive designs.

II. ARCHITECTURAL METHODOLOGY

In many ways, the architecture and organization of a VLSI processor are similar to the designs used in the CPUs of modern machines implemented by using standard parts (TTL, ECL, etc.). The design methodology used to optimize the design and deal with complexity are thus also similar. The MOS technology imposes some new constraints that emphasize the interaction between architecture and implementation. This forces the architectural designer to be more aware of the implications of his decisions.

A computer architecture should be measured by its effectiveness as a host for applications and by the performance levels obtainable by implementations of the architecture. For a general-purpose processor, the suitability of an architecture as a host is largely determined by its effectiveness in supporting high-level languages. A special-purpose machine can be thought of as an architecture that supports a restricted class of languages and applications. For example, a special-purpose chip for graphics applications, such as the Geometry Engine [1], handles a restricted input language describing geometric transformations. The nature of the language and the required performance for primitives in the language is dictated by the structure of the applications. In other cases, a very general language is appropriate as the programming language for the chip, but the intended application skews the frequency of various operations in the input or mandates additional performance constraints. A special-purpose signal processor may fall in this class: numeric operations occur with higher-than-usual frequency and are time critical.

In the next sections, we discuss the issues that arise in determining the suitability of an architecture as a program host and the implications of the architecture on the organization and summarize by proposing some guidelines to help evaluate the suitability of an architecture both for an application environment and for implementation using VLSI.

A. Architecture as Program Host

The efficiency of an architecture must be evaluated both on the cost and on the performance of implementations of that architecture for programs of interest. This evaluation must make realistic assumptions about the use of a

programming language and the class of applications of interest. Since most programming is done in high-level languages, performance benchmarks must be based on measuring such programs; benchmarks based on assembly language performance are not very useful because they do not accurately measure high-level language performance.

To measure the effectiveness of an architecture for executing high-level language (HLL) programs requires an HLL compiler for the proposed architecture. The quality and structure of the HLL compiler affects the architectural measurements. For example, whether or not the compiler registers allocation, or global optimization, can dramatically alter instruction profiles. Some architects advocate eliminating or neutralizing the effect of the compiler, usually by assuming a naive compiler technology. This design approach is flawed for two reasons. First, these measurements will not reflect the correct design decisions for an environment with better compiler technology. For example, if a compiler without register allocation is used, the architect may conclude that the machine should have only a very small number of registers, since they are not heavily used. A second, and more subtle, flaw is that the architecture influences the difficulty of building compilers. To effectively utilize some architectures, especially those with sophisticated and powerful instructions, requires a potent compiler. It may not be possible to effectively utilize some of these features, or it may require compilation techniques that are not acceptable in practice due to high compilation cost. Without actually constructing a compiler, it is difficult to measure these effects.

By correctly using the compiler we can arrive at a reasonable approach to making trade-offs in the instruction set design. Let us assume that the basic structure of the instruction set is in place. This structure is determined by the expected programming languages, the applications, the implementation issues, and the general state of compiler technology. The starting point must include a reasonable list of operators, some addressing modes, and a set of rules for combining these two. The compiler can then be used as the evaluation tool for different architectural features.

Consider the following scenario that might arise in many instruction set design processes. A new set of addressing modes is proposed for the architecture. The addressing mode additions will require that instructions be lengthened to accommodate a longer operand format. From the compiler viewpoint we can evaluate the worth of these additional addressing modes by comparing the quality of the code generator both with and without such architectural features. To make this evaluation, the primary piece of data needed is the execution time (in instruction counts) for the alternatives; other useful pieces of data include the frequency of use of the new addressing modes and the difference in dynamic instruction bandwidth. These second-

ary data are useful to corroborate the instruction count data and to understand other effects of the additions to the instruction set. A similar measurement process might be used when considering the addition or replacement of an opcode. In either event, to complete the evaluation of the additions to the architecture we need to evaluate the additional implementation cost; we discuss this impact in more detail in the next section.

It is insufficient to consider only the needs of a compiler in designing an architecture: an operating system is required to make the hardware useful for executing applications. The operating system requires certain architectural capabilities to achieve full functional performance with reasonable efficiency. If the necessary features are missing, the operating system will be forced to forego some of its user-level functions or accept significant performance penalities that may make the architecture unacceptable. Some designers have advocated that the architecture provide special support for certain operating system functions.

Such architectural features should be subject to the same evaluation process used when examining instruction set changes. Of course, such measurements are more difficult because they involve an operating system rather than just a compiler. Estimates based on existing operating systems and their execution may be the only reasonable method to obtain data that measures the use of operating system support features contemplated for inclusion in the architecture. Utilization data from the operating system together with estimates of operating system execution time (as a percentage of all execution time) can be used to estimate the architectural performance gain over a design without such special support. Of course, when considering such features we must also evaluate their implementation cost.

B. Architecture as Implementation Requirements

The structure of an architecture dramatically affects the type of organization needed to obtain certain performance levels from an implementation of that architecture. Likewise, given a framework for an implementation, the ability of the implementation to support different architectural features will vary widely. This is especially true when the implementation is in VLSI, in which the interaction of the architecture and its implementation is more pronounced. Three key properties of the technology are important and tend to accentuate these interactions. These are the basic gate switching speeds, communication costs, and the effects of chip boundaries.

The MOS technology sacrifices speed for density; this encourages the use of parallel implementations. That is, many slower components are used rather than a smaller number of fast components. This basic design method-

ology has been the key leverage in a number of projects as varied as systolic arrays [2] to the Micro VAX-I data-path chip [3]. In the case of systolic arrays, the individual processors can be quite slow because overall performance is based on utilization of parallel, pipelined hardware units.

As a general rule, communication is more expensive than computation. Architectures that require significant amounts of global interaction will suffer in implementation. Thus, the architect should arrange to make use of local communication and computation whenever possible. The chip boundaries have two major effects. First, they impose hard limits on data bandwidth on and off the chip. Second, they create a substantial disparity between on-chip and off-chip communication delays. This partitioning of the environment into on-chip and off-chip objects is particularly important. The designer must use on-chip resources to cache the global environment and lower off-chip communication requirements. The size and complexity of the architecture can force the designer to partition the system in a suboptimal manner and face very high interchip communication cost.

Another factor that depends on both the architecture and chosen organization is the complexity of the design. Complexity is a factor affecting both design time and performance in any implementation medium, but is exacerbated in VLSI, in which complexity becomes more difficult to accommodate. The costs of debugging and iterating on the design make complex designs extremely difficult to complete in a reasonable time. The process of performance tuning is also restrained by the complexity of the basic design. With limited resources to complete a complex design, the implementors will have to sacrifice performance in an attempt to get a functional chip. The most important design corollary from this observation is that no architectural feature comes for free. At the least, it complicates the design, increasing design time, and often decreasing clock-speed performance, if not actual functional performance.

The architecture primarily affects the performance of the hardware at the organizational level, where it imposes certain requirements. Smaller effects occur at lower implementation levels at which the technology and its properties become relevant. The technology acts strongly as a weighting factor that favors some organizational approaches and penalizes others. It is useful to examine our architectural trade-off scenario from the implementation viewpoint. The instruction count data gave us an architectural measure of the value of the proposed extensions. At the implementation level, the key issue becomes the clock speed that is obtainable with the two architectural alternatives. Together, the architectural and implementation measurements gave a good picture of whether the feature belongs in the architecture. Of course, it is more difficult to measure the implementation cost, since it is probably impractical to design the chip with all possible proposed features. Experienced designers may be the best estimators of such costs.