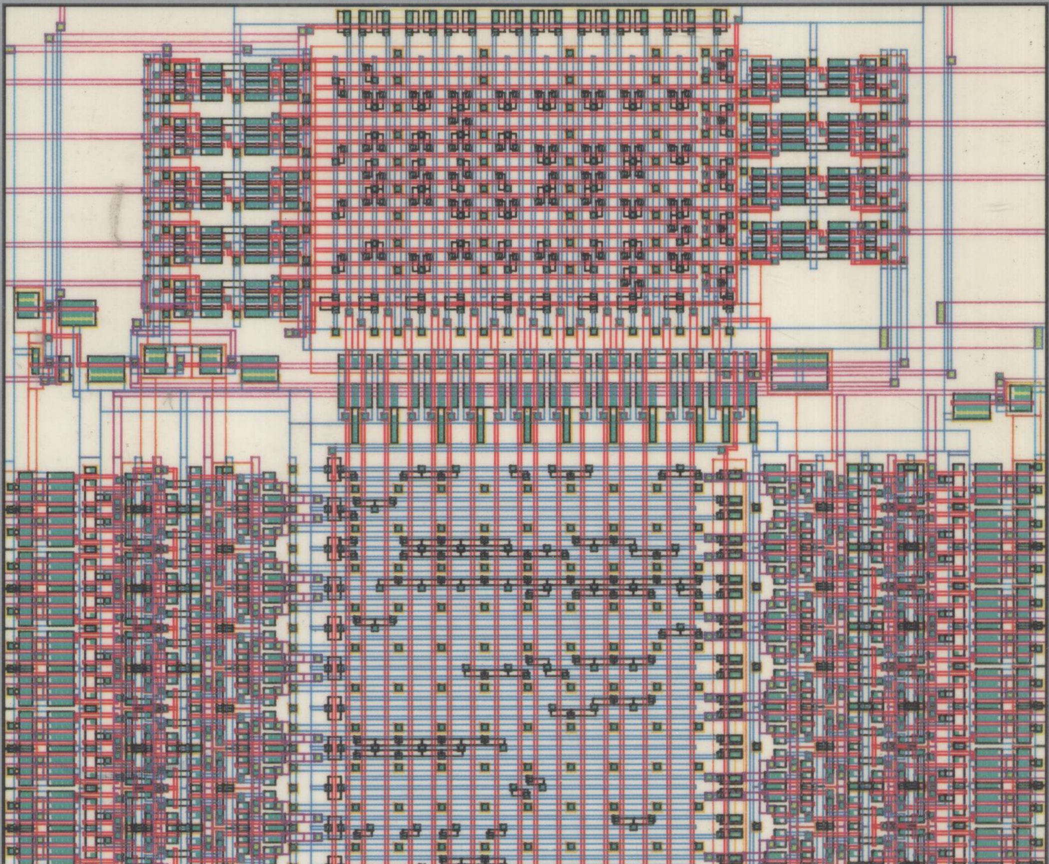
E. Hörbst · C. Müller-Schloer · H. Schwärtzel

Design of VLSI Circuits

Based on VENUS



Springer-Verlag

E. Hörbst · C. Müller-Schloer · H. Schwärtzel

Design of VLSI Circuits

Based on VENUS

With 192 Figures

Springer-Verlag Berlin Heidelberg New York
London Paris Tokyo 1987

Dr. phil. Egon Hörbst
Managing Director of the CAD Project "VENUS", Corporate Applied Computer Sciences, Siemens AG, Munich

Dr. Ing. Christian Müller-Schloer
Group Leader "Systems Architectures", Corporate Applied Computer Sciences, Siemens AG, Munich

Prof. Dr. techn. Heinz Schwärtzel
Managing Director of Corporate Applied Computer Sciences, Siemens AG, Munich

ISBN 3-540-17663-2 Springer-Verlag Berlin Heidelberg New York
ISBN 0-387-17663-2 Springer-Verlag New York Heidelberg Berlin

Library of Congress Catalog Card Number.
Hörbst, Egon. VENUS, design methods for integrated circuits.
Bibliography: p. Includes index.
1. Integrated circuits – Design and construction – Data processing. I. Müller-Schloer, C. (Christian), 1950– .
II. Schwärtzel, H. III. Title. TK7874.H673 1987 621.381'73 87–9624
ISBN 0-387-17663-2 (U.S.)

Typesetting: Triltsch, Würzburg; Printing: Ruksaldruck, Berlin; Bookbinding: Lüderitz & Bauer, Berlin
2362/3020-543210

Preface

Microelectronics are certainly one of the key-technologies of our time. They are a key factor of technological and economic progress. They effect the fields of automation, information and communication, leading to the development of new applications and markets.

Attention should be focused on three areas of development:

- process and production technology,
- test technology,
- design technology.

Clearly, because of the development of new application fields, the skill of designing integrated circuits should not be limited to a few, highly specialized experts. Rather, this ability should be made available to all system and design engineers as a new application technology – just like programming technology for software.

For this reason, design procedures have to be developed which, supported by appropriate CAD systems, provide the design engineer with tools for representation, effective instruments for design and reliable tools for verification, ensuring simple, proper and easily controllable interfaces for the manufacturing and test processes. Such CAD systems are called standard design systems. They open the way to fast and safe design of integrated circuits.

First, this book demonstrates basic principles with an example of the Siemens design system VENUS, gives a general introduction to the method of designing integrated circuits, familiarizes the reader with basic semiconductor and circuit technologies, shows the various methods of layout design, and presents necessary concepts and strategies of test technology.

Second, the book imparts all information necessary for designing integrated circuits with a standard design system, including essential user instructions for VENUS in a simplified form. In addition, the entire design process for an integrated circuit is described with an example of a simple circuit serving as a learning tool. Moreover some of the cell libraries which are offered as part of VENUS are listed, and a choice of cells sufficient for one's own designs are described in detail as a cell catalogue. Methods of cell library development and quality assurance are also presented. Finally, this book shows developing trends and the possibility of placing design systems for integrated circuits within the framework of CAD systems used for developing larger types of electronic systems.

The drive to develop standard design systems comes from the field of systems technology which has also helped to shape their current capabilities. The development of the design system VENUS is based upon the entire range of electronic systems technology such as computer technology, whose requirements were pointed out by Dr. Braeckelmann, communications technology – whose requirements were pointed out by Dr. Pfrenger, Dr. von Sichart, Dr. Stegmeier –, and automations technology – whose requirements were pointed out by Mr. Schaeff and Mr. Dittmann.

It is particularly important for standard design procedures that the manufacturer guarantees reliable operation and correctness of cell libraries. This is possible only if the libraries and processes are developed with careful quality control. This was done by Mr. Saehn, Dr. Schrader, and Dr. Zibert from the Component Division.

Further contributions to single chapters of the book were provided by: M. Gonauser, Dr. A. Gilg, M. Hernandez and K. Forster. The authors owe special gratitude to Ms. E. Pfeuffer, Mr. M. Spaventa and Mr. M. Müller who were responsible for the English version of the book. Ms. B. Fruehauf, Ms. K. Brosseder and Ms. C. Zeiser rewrote the text several times with great patience.

Munich, April 1987

E. Hörbst
C. Müller-Schloer
H. Schwärtzel

List of Co-Authors

Paul Birzele	Chapter 3
Manfred Gerner	Chapter 4
Ernst Göttler	Chapter 5
Otto Grüter	Chapter 6
Karlheinz Horninger	Chapter 2
Gerd Sandweg	Chapter 7
Thomas Wecker	Chapter 3

All co-authors are members of Corporate Applied Computer Sciences, Siemens AG, Munich.

Contents

1 Introduction to the Design of Integrated Circuits

This book has three main goals:

First, we want to show that the development of modern design procedures for integrated circuits which have simple and formal interfaces with the design engineer as well as with the production process (standard design procedures) have resulted in new ways of solving user-specific problems: A new kind of application technology has thus been formed. It might find as wide an application as the programming technology for software which, in the beginning of the sixties, became a universal application technology. Second, we want to give general introduction to the methods of designing integrated circuits. We will discuss basic technologies as well as various methods of layout design and required testing concepts.

Third, we want to familiarize the user with the simplified methodology of designing integrated circuits with a standard design procedure. VENUS [1] serves as a learning tool for explaining the current level of desgin automation. The book is, however, not meant as a substitute for a detailed VENUS user manual, nor is it just a catalogue of the functions and cells that any user manual contains.

Chapters 1 to 4 are meant for the reader who wishes to obtain a summary of design methods for integrated circuits, technologies for microelectronics, various layout design methods and the aspects of testing.

Chapters 5 and 6 impart the practical know-how required for designing integrated circuits with VENUS. They also include basic instructions for using VENUS. All steps, starting with the preparatory considerations necessary for each part of the design up to the test, are described and illustrated with an example used as pedagogic tool. Furthermore, Chap. 5 describes the full range of cell libraries and masters offered in VENUS 1; it deals with the problems of developing them and assuring their quality, and it offers a choice of cells arranged in a cell catalogue adequate for user designs.

Finally, in Chap. 7, VENUS is placed in the wider scope of designing large electronic systems.

1.1 User Technology for Digital Electronics

During the last 30 years digital technology has, to an incredible extent, been successful in solving problems arising within applied electrical engineering. The most outstanding and common example in the field of digital electrical engineering is the processor

[1] Registered trademark of Siemens AG, Munich. The name is the acronym for "VLSI-Entwicklung und Simulation" (VLSI development and simulation).

which forms the heart of the computer. Together with the computer, a new user technology was developed in the early sixties, namely, programming technology for software. Packages of methods and tools were developed: The *software (SW) programming systems.*

Their main components are: the programming language, the compiler and a package of tools for specification, editing, test and documentation. More recent developments combine all these components, thus creating programming environments.

The scope of the computer, supplemented by other digital circuits in order to reach specific system solutions, is constantly expanding into other fields of technology, such as testing technology, automatic control technology, communications technology, office technology and many others. In order to explain this phenomenon, one needs only look at the advantages of digital technology in terms of reliability, adaptability and efficiency in problem solving, due to progress in the field of circuit integration.

The standardizing effect digital technology has had on the methods of design has probably been a decisive factor for the spread of digital circuits. The components of SSI [2] circuit families in TTL [3] or CMOS [4] technologies has influenced a whole generation of circuit designers. It was no longer the level of distinct individual transistors (the smallest analog units) that formed the base of the design, but rather the complex, digital level of logic gates, flipflops, counters and decoders. With that, a building block system has been created which allows the solving of customer-specific problems by means of free configurations formed quickly from standard components.

For this purpose, the circuit designer had to design a circuit on gate level, based on a specification. It was possible for him to make use of the standard circuits offered by the manufacturers of integrated circuits (ICs). The subsystems built on printed circuit boards (pc boards) were finally combined in a customer-specific system solution (see Fig. 1.1). With about ten equivalent circuit functions per IC and thirty ICs per board, a circuit with 1500 gates thus required approximately five boards!

With increasing complexity of integration, complete subsystems with several thousands of transistors could be placed on one component or chip. For IC manufacturers the problem arose that the specialization, owing to increased complexity, was reflected in smaller production volumes per type. This specialization had to be reconciled with the demand for a high number of pieces, raised by the production lines. The solution was to develop generally applicable standard components, the most prominent type of which finally became the microprocessor. With the microprocessor, that part of a circuit which is application-, or customer-specified could be shifted back to the software.

From the very beginning, of course, the microprocessor was developed for a group of users different from that of computers. The group which uses the microprocessor is that of system engineers. Together with the microprocessor, a second, new application technology for electronics was developed in the second half of the seventies, whose tools were combined in microprocessor (μP) development systems. Their major

---

[2] SSI: Small Scale Integration.

[3] TTL: Transistor Transistor Logic.

[4] CMOS: Complementary Metal Oxide Semiconductor.

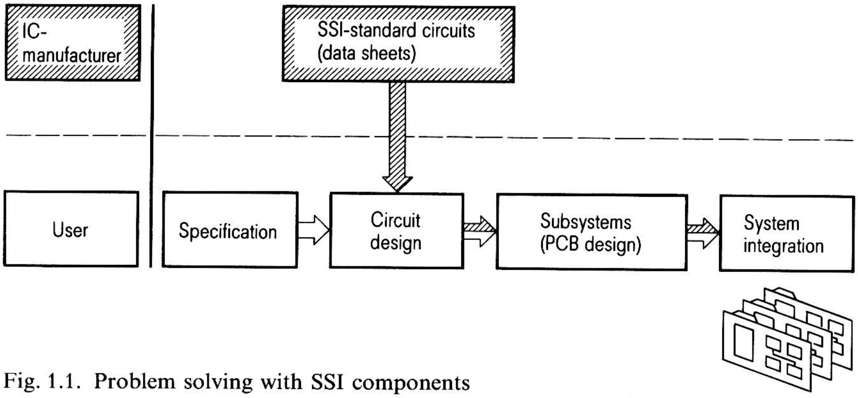


Fig. 1.1. Problem solving with SSI components

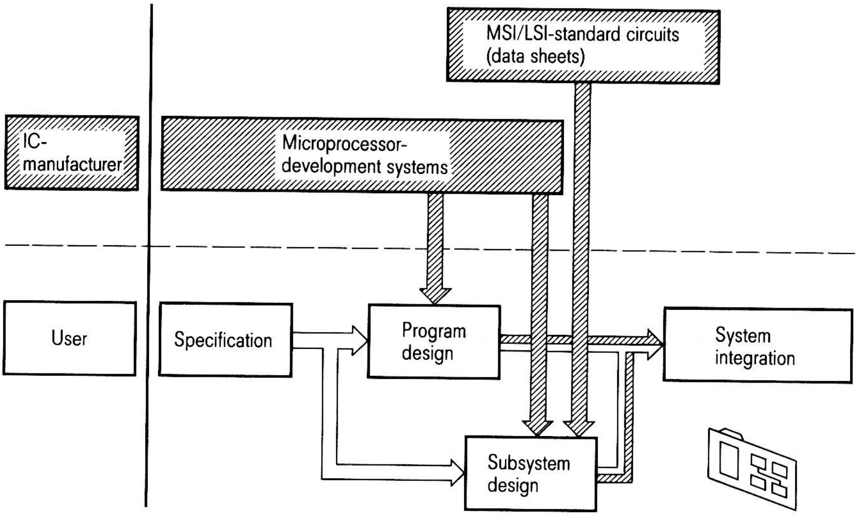


Fig. 1.2. Problem solving with a microprocessor and MSI/LSI components

components are a SW programming system, described above, and tools for hardware/software integration and testing.

With this expansion of integration technology, the IC manufacturer now had to fulfill the larger tasks of developing, manufacturing and supporting the entire package of software and hardware development as well as system integration and test. This resulted in the microprocessor development system described above.

Thus, the development process can be carried out as follows:

From the variety of processors, peripherals and memories, the user chooses the appropriate components, combines them on a board in a suitable way, and finally programs his computer in order to obtain the customer-specific system solution (Fig. 1.2). If standard single-board computers are used, the user's work is limited to the generation of the software. In the following, this type of solution is called "software (SW) method" in microelectronics. The result is a program that can be loaded in a computer.

Increased demand for compactness, reliability, capacity and cost efficiency finally led to an increased number of users wishing to integrate their circuits on their own chip ("custom chip"). This, however, was at first impeded by the comparatively high development cost. Because of the unusually low number of pieces produced, the unit cost of this kind of chip was too high. The solution was found in comprehensive formalization which allowed an automation of the design procedure for integrated circuits. Although an IC design procedure requires a variety of time-consuming and interdependent steps, a modern design system provides a simple standard procedure which the user can learn to manage within a short period. On one hand, tools for the individual steps of design are available for use in such design systems. On the other hand, the organizational process which has become more complicated, is supervised and automatically adhered to. For developing chips which are specifed by their users ("personal chips"), a third application technology of microelectronics was formed in the early eighties. Its tools are combined in *IC design systems*.

In the following, the application of IC design systems is called "*hardware* (HW) *method*" in microelectronics. The result of a development process that follows this method is a hardware component: more precisely, the production and test documents for a hardware component, a chip. The description of the hardware method is the content of this book. The basic approach, combined with the intensive interaction between IC producer and user, is shown in Fig. 1.3.

When the Figs. 1.1 to 1.3 are compared, it becomes clear that, with increased integration, wider scope of application, and higher efficiency of integrated circuits, the IC manufacturer was increasingly forced to make necessary user technologies available.

The rise of the three standard application technologies in the field of digital electronics is shown in Fig. 1.4.

A comparison of the two more recent user technologies which are based upon microprocessor development systems (SW method) and IC-design systems (HW

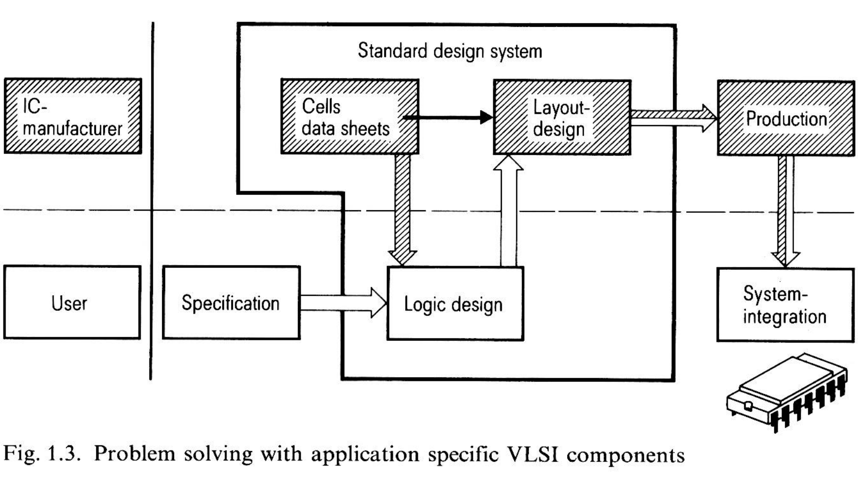


Fig. 1.3. Problem solving with application specific VLSI components

<table>
<tr><th colspan="3">Technological medium</th><th>System support for application technology</th><th>Date of origin</th></tr>
<tr><td rowspan="3">Digital circuit</td><td colspan="2">Processor</td><td>SW programming system</td><td>Since 1963</td></tr>
<tr><td rowspan="2">Inte-grated circuit</td><td>Micro-pro-cessor</td><td>µP development system</td><td>Since 1973</td></tr>
<tr><td>Applica-tion specific IC</td><td>IC design system</td><td>Since 1983</td></tr>
</table>

Fig. 1.4. Application technologies of digital electronics

method), shows interesting analogies. For this reason, the development processes for one problem, utilizing the different technologies, are compared in the following two paragraphs.

This comparison shows some important similarities, but also a number of distinctions, resulting in a better understanding of the special features of the respective development process.

Both alternatives have the basic methods of problem solving in common. Those methods will be discussed in more detail in the following paragraph. Thereafter, SW method and HW method will be outlined, showing the individual steps, and compared by means of an example.

1.2 The Process of Problem Solving

In connection with microelectronics development, two new user technologies have arisen for systems development. For problem solving with microprocessors, the microprocessor development systems are available. This kind of problem approach is called "SW method". For problem solving with problem-specified integrated circuits, IC design systems have been developed. This way of problem solving is called "HW method". A comparison of the two approaches makes clear that, at least in this rough sketch, there are no differences in principle between SW method and HW method.

1.2.1 Phase Model of the Problem Solving Process

The description of every complicated process requires the framework of a basic model. For this reason, we first want to outline a basic model of the problem solving process on an abstract level.

Today, it is common to use a phase model as the basic model for complex system development. For organizational reasons, this is done mainly in order to divide the process of problem solving into controllable sections, where results can simply be stated, and tasks can be distributed.

Because of computer aided automation, this is mainly done to obtain exact interfaces for the flow of information, and to ensure the consistency of the data. The phase model is divided into six phases.

The starting point of the problem solving process is always an exact verbal description of the requirements (requirement catalogue). This is obtained as the result of a *study phase* (1).

Based upon this study, the specification or performance description is developed by applying available techniques for problem solving. The description consists of two components, object specification and test specification. The object specification includes the solution concept, the latter consisting of the functional concept, the performance concept and the structure concept.

For the development of subsystems, their arrangement in the main system plays a crucial role. The interface between subsystem and main system is defined by criteria such as type, format and range of data and command flow. For object specification, the performance requirements are also defined. These are, in most cases, the determination of critical time conditions, or of maximum values, such as the maximum current supply or the size of memory.

Test specification, a tool which becomes more and more important as problem solving in electronics becomes increasingly complex, describes test methods, test tools, cases and processes.

The specification is the result of the specification phase (2).

Now the realization or actual development begins with the design phase or, as in this case where complex problems have to be solved, the *system design phase* (3). Starting with the specification, a more formalized description or representation, is begun. Usually a top-town method is applied. The system function is divided into hierarchical levels from top to bottom. It is divided into single functions which are placed in a static connection structure, and whose interactions are controlled by means of a dynamic cooperation structure. Connection and cooperation structures are called function structures. In autonomous processing systems, however, the cooperation structure often consists in turn of two components: one for the information flow, and one for the control flow.

The subsystems are designed in more detail in a step-by-step resolution process. At each level of this process, the target function of the subsystem is divided into a number of subfunctions which are represented by function blocks (modules), and are connected to each other through a functional structure. The flow of control commands which integrates these subfunctions according to the functional structure of the subsystem, can be described with procedural diagrams, for example, in the form of Nassi-Schneidermann diagrams. The hierarchically layered resolution process is ended when the elementary functions have been reached.

In this phase of the development process, the corner-stones are set for an organization which, often necessarily, divides the task of succeeding design steps, that is, it allocates the individual subfunctions of the subsystems to parallel working groups. In this phase, the complete interface architecture is also developed. The test system is

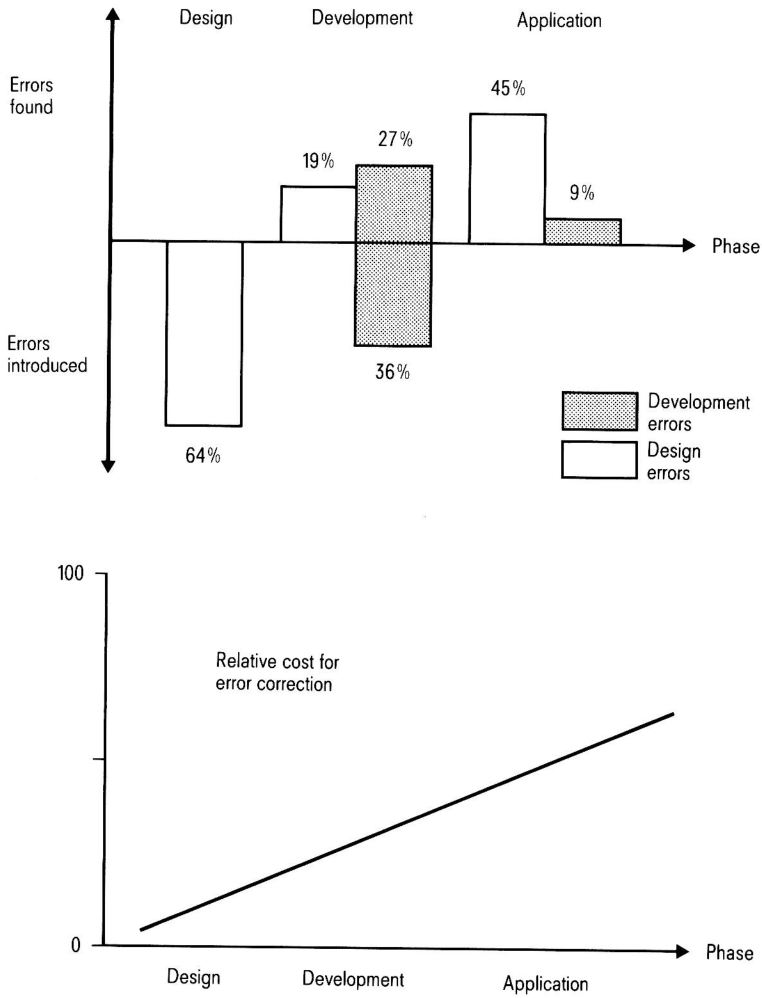


Fig. 1.5. Program errors and debugging expenditure

also prepared which must allow for the individual test (module test) as well as the system test (integration test).

In the system design phase, far-reaching assessments are made, such as the formal representation of system functions (target functions), division into subfunctions by hierarchically layered resolution, design of interface architecture, connection structure and functional structure. For complex problems such decisions cannot, in general, be made with certainty. Wrong decisions and design errors in this phase usually cause especially undesirable effects: they are often discovered very late and then require a disproportionately high amount of repair – if repair is possible at all. Systematic observations during SW development led to recognition of the correlation between error frequency and debugging effort within each phase of the problem solving process (Fig. 1.5).