


International Lecture Series in Computer Science

*The International Chair in Computer Science was  
created by IBM Belgium in cooperation with the  
National Foundation for Scientific Research*

# Algorithmics for VLSI

Edited by  
**C. Trullemans**

 Academic  
Press

TN47  
7866

8860855

# Algorithmics for VLSI

*Edited by*

**C. Trullemans**

*Laboratoire de Microelectronique  
Université Catholique de Louvain  
Louvain-La-Neuve  
Belgium*



1986



E8860855

**ACADEMIC PRESS**

*Harcourt Brace Jovanovich, Publishers*

London Orlando San Diego New York Austin  
Boston Toronto Sydney Tokyo

ACADEMIC PRESS INC. (LONDON) LTD.  
24-28 Oval Road  
London NW1 7DX

*U.S. Edition published by*  
ACADEMIC PRESS INC.  
Orlando, Florida 32887

Copyright © 1986 by  
ACADEMIC PRESS INC. (LONDON) LTD.

*All Rights Reserved*

No part of this book may be reproduced in any form by photostat, microfilm, or any other means, without written permission from the publishers

British Library Cataloguing in Publication Data

---

Algorithmics for VLSI

**ISBN 0-12-701230-3**

Photoset by Paston Press, Norwich and printed by  
St Edmundsbury Press Ltd., Bury St Edmunds, Suffolk

## Algorithmics for VLSI



## **International Lecture Series in Computer Science**

These volumes are based on the lectures given during a series of specially funded chairs. The International Chair in Computer Science was created by IBM Belgium in co-operation with the Belgium National Foundation for Scientific Research. The holders of each chair cover a subject area considered to be of particular relevance to current developments in computer science.

The Correctness Problem in Computer Science (1981)  
R.S. BOYER and J. STROTHER MOORE

Computer-aided Modelling and Simulation (1982)  
J.A. SPRIET and G.C. VANSTEENKISTE

Probability Theory and Computer Science (1983)  
G. LOUCHARD and G. LATOUCHE

New Computer Architectures (1984)  
J. TIBERGHIE

Algorithmics for VLSI (1986)  
C. TRULLEMANS

# Contributors

---

**F. Anceau** BULL Systèmes, Les Clayes St Bois, F-78340 France

**R. M. Lea** Brunel University, Uxbridge, Middlesex UB8 3PH, England

**Th. Lengauer** FB 10, Universität des Saarlandes, D-6600 Saarbrücken,  
Federal Republic of Germany

**K. Mehlhorn** FB 10, Universität des Saarlandes, D-6600 Saarbrücken,  
Federal Republic of Germany

# Preface

---

The first integrated circuit was developed in July 1958 by J. Kilby. This was an equally important step in the history of both computer science and electronics. It was also the starting point of an evolution towards a field of common interest for computer scientists and electronic engineers: algorithmics for very large scale integrated circuits.

Bridging the gap between algorithmics and circuits is possible thanks to the level of maturity reached today by integrated circuit technology. The introduction of FORTRAN compilers, operating systems, and transistor logic circuits used in computers began in 1958.

As compared to vacuum tubes devices, transistor circuits exhibit a drastic reduction in size and power requirement, and a corresponding increase in reliability. The major problem in attempting a further size reduction is to develop a better interconnection technology. This is basically the aim of integrated circuit technology. This problem was of prime importance in the late 1950s and early integrated MOS logic networks were patented in 1957.

Similarly, the need for a telephone exchange system without any moving mechanical part was the driving force behind the quest for the bipolar transistor, discovered by Shockley, Brattain and Bardeen, who were awarded the Nobel prize in 1949. The discovery of the bipolar transistor is an example of successful interaction between formerly separated fields. Shockley, Brattain and Bardeen had been taught solid state physics by Schrödinger, Franck, Sommerfeld and Schottky, who were among the founders of this new science; through them, a direct link was established between the fundamental research of the physicists in the 1930s and the integrated circuit by Kilby at the end of the 1950s.

The well-known TTL series is another step in the same direction. It is a set of electronic components. However, the behaviour of TTL components is mostly described using Boolean equations: they are a materialization of abstract mathematical entities, ideal logic operators. This way of looking at them is far removed from solid state physics. However, a TTL circuit is built in such a way as to allow this abstract description of its behaviour, without any care for the electrical details. A TTL circuit is, nevertheless, fairly simple. Progress in fabrication technology has allowed for much more powerful primitives to be realized. A chip currently contains several tens of thousands of transistors.

To organize such a complex system, or to write software on such a large scale, are similar problems. Coming from simple circuit to complex systems,

electronic engineers had to join computer scientists in translating onto chips the architecture of minicomputers.

The celebrated – and debated – book, *Introduction to VLSI Systems* by C. Mead and L. Conway, is a sign of a corresponding interest from the computer scientist community. This book is merely an indicator of the emergence of this common interest in VLSI.

The catalog of TTL components was the marker of a previous meeting point. A large set of machines have been built from this restricted set of primitives. The possibilities opened by the VLSI are, however, much more fascinating.

An algorithm handles a set of data in order to produce a solution. Choosing the better method is merely a trade-off between the cost and the performance of the solution. The context of VLSI is, however, very different from the context of classical machines, and this remark opens the way for a whole class of new algorithms.

The new possibilities offered to the computer by the VLSI will obviously change the way computers are constructed, and they will also change what computers are doing. There is obviously a large qualitative difference – not only a quantitative one – between the computation of mathematical tables, which was the task of the first computers, and the job of an office automation system.

Integrated circuit technology is naturally oriented towards large volume production. It may be compared to printing works. An observer of the 15th century, even if able to forecast the unbelievable quantity of printed paper which is distributed today, would probably not have foreseen the consequence of this flow of texts on a now well-read mankind. Our own ability to forecast is no better. We can only guess that an increasing quantity of algorithmic machines will spread everywhere. At least one may conjecture that the undue standardization issued by the industrial revolution can be cured thanks to information technology.

At the present time, however, standardization is even an internal law of VLSI technology itself: general purpose microprocessors and memories share the larger part of the market. However, custom chips are gaining in importance; they could reach 90% of the market at the end of the 1980s. This is a fundamental trend, explained by the maturation of design techniques, and by the evolution of the processing equipment. A process line is more and more crowded by robots, able to modify their behaviour when facing new problems. A classical bonding machine, driven by cams, can only handle standard chips. Any modification to the bonding-pattern asks for costly mechanical adjustments. An intelligent robot, driven by a pattern recognition system, can handle custom chips at no extra cost.

Moreover, chips themselves are just part of the story: the largest producer

of integrated circuits in the world is IBM; however, IBM does not sell integrated circuits, but computers. A VLSI is merely a component of a larger system; the full power of VLSI is to make feasible complex systems at low cost. Even if internally complex, these systems may externally look quite simple. In this way, many problems have a VLSI solution, making full use of the intrinsic capabilities of VLSI technology.

Part 1 of this book (Prof. Lea) introduces the basic concepts of VLSI architectures. Original implementations of special purpose architectures are then described, including a performance comparison between several design styles (e.g. single instructions single data – SISD, and single instruction multiple data – SIMD). In part 2 (Profs. Lengauer and Mehlhorn), a theoretical model for complexity for VLSI is developed and applied to the design of efficient VLSI algorithms. The HILL design system, including a symbolic layout editor and a switch level simulator are also described. Part 3 (Prof. Anceau) is devoted to a detailed analysis of layout design styles. This analysis leads to the description of processor templates, well suited for silicon compilation (part 4). Part 5 summarizes the evolution of processor architecture up to the VLSI era.

The design of complex VLSI systems is adequately performed by borrowing techniques commonly used in computer science, and conversely the versatility of the VLSI is such that it allows for an efficient implementation of complex functions. The algorithmics for VLSI is a promising domain of research, and is likely to be added as a new chapter to the core of computer science. The contribution of the authors of this book is therefore especially welcomed in this series which is devoted to subject areas of particular relevance to computer science.

This book covers the material of a series of lectures given at Louvain-la-Neuve during the academic year 1982–83. The contributors to this book were the holders of the international professorship in computer sciences, organized by the “Fonds de la Recherche Scientifique”, which is promoting high-level research and education in Belgium. IBM Belgium is generously funding this professorship. I am pleased to express my gratitude to these organizations, to which the attendees to the lectures and the readers of this book are indebted also. I feel also sincerely thankful to the people who helped in the organization of this series of lectures, and especially to Mrs M. Mercenier and Mr M. Windael.

CH. TRULLEMANS

# Contents

---

Contributors . . . . .	v
Preface . . . . .	vii

**1. VLSI parallel-processing chip architecture**

R.M. Lea

<b>1. Introduction to chip architecture . . . . .</b>	<b>1</b>
1.1. VLSI chip architecture . . . . .	2
1.2. VLSI parallel-processing chip architecture . . . . .	8
<b>2. Non-numerical information processing . . . . .</b>	<b>11</b>
2.1. Associative string processing . . . . .	12
2.2. Associative string processing: SISD implementation . . . . .	13
2.3. Associative string processing: SIMD “associative parallel processor” implementation . . . . .	15
<b>3. Digital signal processing . . . . .</b>	<b>19</b>
3.1. Linear filtering . . . . .	21
3.2. Spatial filtering . . . . .	26
3.3. The SCAPE chip . . . . .	29
<b>References . . . . .</b>	<b>31</b>

**2. VLSI complexity, efficient VLSI algorithms and the HILL design system**

Th. Lengauer and K. Mehlhorn

<b>1. Introduction . . . . .</b>	<b>33</b>
<b>2. A complexity theory for VLSI . . . . .</b>	<b>34</b>
2.1. The VLSI model . . . . .	34
2.2. Communication complexity . . . . .	37
2.3. Extensions and related results . . . . .	44
<b>3. Efficient VLSI algorithms . . . . .</b>	<b>46</b>
<b>4. The HILL design system . . . . .</b>	<b>50</b>
4.1. HILL layout language and graphics editor . . . . .	50
4.2. Compaction in HILL . . . . .	58
4.3. HILLSIM, a switch-level simulator . . . . .	71
<b>5. Recent results . . . . .</b>	<b>83</b>
<b>References . . . . .</b>	<b>87</b>

**3. Statistical properties and layout strategies for NMOS and CMOS layout**

F. Anceau

<b>1. Introduction . . . . .</b>	<b>91</b>
----------------------------------	-----------

<b>2. Cell deformability</b>	93
<b>3. Wiring strategy</b>	93
<b>4. NMOS/HMOS technologies</b>	96
4.1. NMOS double-poly and double-metal technologies	96
<b>5. Transparency function</b>	98
<b>6. Layout density for transistors</b>	99
<b>7. CMOS technologies</b>	99
7.1. CMOS single-poly and single-metal technologies	99
7.2. CMOS double-poly and double-metal technologies	101
<b>8. A tool for floor planning</b>	102
<b>9. Conclusion</b>	103
<b>References</b>	103

**4. Silicon compilation for microprocessor-like VLSI**

F. Anceau

<b>1. Introduction</b>	105
<b>2. General considerations</b>	106
<b>3. Data-path design</b>	106
3.1. Data-path templates	106
3.2. Data-path compilation	108
3.3. Parallelism into data path	109
3.4. Access to subfields of registers	110
3.5. Data-path layout	111
<b>4. Control-section design</b>	112
4.1. General considerations	112
4.2. Using the notion of function slices	112
4.3. Control-section compilation	114
<b>5. Conclusions</b>	114
<b>References</b>	115

**5. LSI-processor architecture**

F. Anceau

<b>1. Introduction</b>	117
<b>2. External and internal computer architecture</b>	118
<b>3. Computer ranges</b>	118
<b>4. Influence of technology on the external architecture</b>	119
<b>5. Influence of technology on the internal architecture</b>	119
<b>6. Data-path and control sections</b>	120
<b>7. Evolution of the internal architecture</b>	121
7.1. Evolution of the layout techniques	121
7.2. First period (1972–1975)	122
7.3. Second period (1975–1980)	123
7.4. Final period (since 1980)	123
7.5. Control-section design	124
<b>8. Conclusion</b>	125
<b>References</b>	126

<b>Index</b>	129
--------------	-----

# **1. VLSI parallel-processing chip architecture**

R.M. LEA

---

## **1 INTRODUCTION TO CHIP ARCHITECTURE**

In less than 25 years the concept of the integrated circuit has matured from 1 transistor on a single silicon chip to approach 1 000 000 transistors per chip; progressing through SSI (up to 100), via MSI (100 to 1000) and LSI (1000 to 100 000) to VLSI (greater than 100 000). A major contribution to this explosive growth has been a remarkable “collaboration” between the microelectronics component industry and the computer manufacturing industry.

The progress of circuit integration can be likened to that of a goods train with a locomotive at each end, representing “technology push” and “systems pull”; the former demonstrating the eagerness of the semiconductor industry to improve its art and the latter demonstrating the reality of market forces. Clearly, rapid progress can be made only when both “locomotives” work in harmony.

Despite occasional periods of slow progress, this dual propulsion system has rushed the “train” along the “rails” connecting SSI, MSI and LSI into the unexplored region of VLSI. At this point (circa 1980), it became clear that VLSI was not just “a little further down the track” and that the coordination of “technology push” (which had made VLSI possible) and “systems pull” (towards smaller, faster and cheaper computers) had become a major challenge.

The availability of SSI packages incorporating a few logic gates (e.g. quad 2-input NAND gates) enabled digital systems engineers to implement logical functions on a single printed circuit board, without understanding the underlying semiconductor technology. Thus the traditional “bottom-up” design style of digital electronics engineering gave ground to a “top-down” style, enabling logicians and programmers to build modular systems, with standard boards, for an expanding computer market.

MSI packages, integrating the functions (e.g. 4-bit ALU) of such boards on a single chip, allowed the simpler construction of more complex computers, with even less emphasis on “bottom-up” design. This evolutionary process continued with the availability of LSI packages, integrating major components (e.g. 8-bit microprocessors and 16K RAMs) on single chips, enabling exclusively “top-down” computer designs, based on the selection of a few highly marketed “plug-in” boards, in an increasingly software engineering environment.

Thus the era of VLSI dawned with the semiconductor industry planning even more sophisticated chips (e.g. 16/32-bit microprocessors and 64/256K RAMs) to provide a source for the computer industry which was eagerly awaiting the next generation of component technology. However, this increase in on-chip complexity brought with it a whole new set of problems. Indeed, the new opportunities offered by VLSI were approached very cautiously by the microelectronics industry. A new “learning curve” had to be climbed, and, because of the speculative nature of VLSI chip development, venture capital was difficult to obtain. Consequently, although the problems and opportunities of VLSI were widely discussed, “VLSI chip projects” remained mainly in the LSI camp! Nevertheless, the prospect of more specific VLSI systems remained sufficiently attractive to stimulate research towards cost-effective solutions for the problems.

Clearly, a watershed had been reached in the evolution of integrated circuits, and new design techniques had to be developed. The management of complexity was the main problem, and so the help of computer scientists was enlisted.

Computer science had already faced problems of complexity in the development of large programs, and hence the well-established concepts of modular, hierarchical, block-structured software development were transferred to VLSI design. Accordingly, a trend towards well-structured (viz regular) “VLSI chip architectures” and formal “VLSI design methodologies” was initiated and the development of software “VLSI design tools” was greatly accelerated. Indeed, VLSI design rapidly became an accepted subject area of computer science. Major national programmes were started and new courses, teaching these VLSI disciplines, sprang up all over the industrial world. Indeed, this book results from one such course.

## 1.1 VLSI Chip Architecture

In contrast to a random collection of transistors, typical VLSI chip layouts (viz floorplans) are partitioned into some ordered configuration of functional blocks, each of which may be composed of a hierarchy of sub-blocks, which are partitioned into some ordered configuration of base blocks. Thus a VLSI

chip floorplan can be regarded as a sort of data structure, representing a “tree” of “composition cells” (viz blocks and sub-blocks) and “leaf cells” (viz base blocks).

Consequently, at the layout level, a VLSI chip architecture comprises a set of detailed “leaf-cell” designs and a strategy of floorplan composition.

#### *1.1.1 Leaf-cell design styles*

There are four clearly identifiable styles of leaf-cell design, which are described below. The styles are ranked in order of decreasing human effort needed for circuit and layout design. This ranking can also be applied to the complexity of leaf-cell testing.

As an aid to understanding, each style is associated with a scientific or engineering discipline and the close analogy between the activities of a chip designer and a gentleman’s tailor is explored.

*1.1.1.1 Full-custom cell.* This is the physicists’ approach. The design is fully optimized to achieve the highest performance, the lowest power dissipation and the smallest non-functional area for the specific requirements of the cell. Although the most intellectually and aesthetically pleasing, this approach usually involves the most intricately detailed and time-consuming circuit and layout design work. Indeed, full-custom leaf-cell design can be likened to the “bespoke” services of the most expensive tailors. In view of the high design costs, full-custom design can only be justified for general-purpose memory and microprocessor chips which can attract a high-volume market (viz greater than 100K devices per year) or for special-purpose devices dedicated to military and space applications in which full optimization is essential and commercial profit is not the most important factor.

*1.1.1.2 Simplified full-custom.* This a more engineering-oriented approach, in which the leaf-cell design is only partially optimized. The designer tries to maximize the use of regular layout patterns (e.g. register and memory blocks, PLAs, decoders, multiplexers, barrel shifters etc.) which can be easily modified to fit the specific requirements of the cell. In this way, the need for intricate design can be limited to those key cells for which full-custom design is cost-effective. Thus the scope for clever circuit design is somewhat restricted and layout work is simplified. Indeed, simplified full-custom leaf-cell design can be likened to the tailoring services of the larger and more exclusive department stores.

The design technique advocated by Mead and Conway (1980) extends the simplified full-custom style to the virtual elimination of circuit design and, by generalizing chip manufacturers’ design rules, to simple manipulation of a

small number of flexible layout structures. Although rather oversimplified for serious chip architects, the Mead and Conway method has undoubtedly encouraged computer scientists to experiment with architectural prototypes.

*1.1.1.3 Semi-custom cell.* This is a more market-oriented approach, in which leaf-cell design is restricted to the layout of simple routing between standard components. Indeed, many commercial ULA (Uncommitted Logic Array), Gate Array and Standard Cell services are now well established. Two styles of semi-custom cell design are evident, these being distinguished by the type of standard component.

(a) *Standard circuit elements.* This is the engineers' approach, since the emphasis is placed on circuit design. Cells comprise bipolar transistors and resistors or organizations of MOS transistors, which can be interconnected to form different circuits. Indeed, this style of semi-custom design can be likened to the "made-to-measure" services of the larger clothing chain stores.

Where layout is restricted to only the interconnection within a fixed placement of circuit elements, batches of preprocessed wafers can be stockpiled for rapid customization. An alternative style allows flexible placement of circuit elements with symbolic routing between them (e.g. stick level design). Such techniques are normally supported with computer graphics, with automatic detection of design-rule violation and layout compaction options.

(b) *Standard cell blocks.* This is the logician's approach, since the emphasis is placed on the logical interconnection of logic gates and functional logic blocks selected from a library of fully engineered standard cells. Indeed, this style of semi-custom design can be likened to the "off-the-peg" tailoring services of the larger chain stores.

Standard layout cells can be regarded as high-level language procedures "compiled in silicon", and hence the standard cell style usually appeals to computer scientists.

*1.1.1.4 Programmable cells.* This is the mathematicians' or computer scientists' approach, since the emphasis is placed on the data content of standard memory-oriented layout blocks. For example, RAM, ROM, CAM multiplexer and PLA blocks can be loaded with data tables for rapid consultation during program execution. Such layout blocks can be used to create very regular chip floorplans which make good use of silicon. Clearly, this style of cell is very well suited to computer generation. Indeed, since the cell design is manifest in the stored data pattern, formal design methods can be employed. In terms of the tailoring analogy, this style can be likened to

the provision of army uniforms; with a “best-fit” or “first-fit” policy, depending on the state of the national economy.

### *1.1.2 Chip floorplans composition styles*

There are five clearly distinguishable styles of chip floorplan composition, which are described below.

*1.1.2.1 Place and route.* This is a “block-oriented” approach to chip composition, for which two sub-classes are evident.

(a) *Random routing.* This style corresponds to the full-custom approach to leaf-cell design, since the leaf cells are placed strategically in the floorplan before interconnection by the shortest possible paths. Consequently, the style leads to random routing patterns, which are time-consuming to layout and test, since they are not amenable to computer assistance. Although, at first thought, such routing seems trivial compared with leaf-cell layouts, it can be very tedious and highly prone to error.

High-volume memory and microprocessor chips, optimized for the highest performance at the lowest cost, usually fall into this category.

(b) *Routing channels.* With this approach, linear arrays of leaf cells are separated by routing channels which interconnect at the ends of the arrays. Usually, the leaf cells are constrained to fit the same vertical dimension and are sited to make the best use of the array length. Cells are then interconnected via the appropriate busses in the routing channels. Clearly, this style, in conjunction with the standard cell style of leaf-cell design, is very well suited to computer generation. Indeed, this combination forms a popular base for commercial “silicon compilers”.

Unfortunately, the simplicity of this composition style is offset by the following problems:

- (i) resiting of cells is often necessary to ease interconnection bottlenecks;
- (ii) poor layout packing density, due to oversized leaf cells and routing channels, leads to cost-ineffective use of silicon;
- (iii) different and unknown path lengths of cell interconnections leads to synchronization problems and poor chip performance.

*1.1.2.2 Route and slot.* This is a “bus-oriented” approach to chip composition, incorporating a much more regular form of routing channel, to ease the problems cited above. Indeed, a uniform routing channel is established on the chip floorplans and then leaf cells are slotted in appropriate locations relative to the channel. Moreover, this style can benefit from the full

advantages of computer-aided design. Two sub-classes of route and slot composition are evident.

(a) *Standard bus*. Popular for microprocessor chip architectures, this style slots leaf cells on either side of a strategically placed bus. Such busses are usually time-multiplexed to save chip area. In conjunction with the standard cell style of leaf-cell design, this style is well suited to computer generation.

(b) *Grid structure*. Currently the most popular routing structure for ULAs and gate arrays, this style is based on an orthogonal grid of routing channels. Leaf cells are slotted in the square “holes” of the grid and connected to the appropriate lines in the channel. Clearly, this style is very well suited to computer generation.

Although considerably easing problems (i), (ii) and (iii) cited above for routing channels, the route and slot style still suffers the following deficiencies:

- (i) much of the active chip area is “wasted” on routing channels;
- (ii) long signal transmission delays, compared with the logic propagation delays of the leaf cells, degrade overall chip performance.

*1.1.2.3 Bit-sliced structures*. This is another “bus-oriented” approach to chip composition, which develops the standard bus style described above to minimize its inherent non-functional chip area and signal transmission delays. The composition style is easily recognized, since data and control signals are routed orthogonally on two separate communication layers (e.g. metal and polysilicon in NMOS technology). Leaf cells are bit-sliced and “hung” below a strategically placed data bus, each bit of the cell aligning with the corresponding bit of the bus. Thus, a standard bus is integrated within the leaf cells, which are fixed in height and line-pitch by the dimensions of the bus.

Although this style is not as well suited to computer generation as route and slot compositions, several silicon compilers are being developed for this approach. Two sub-classes of bit-sliced composition are evident.

(a) *Abutting cells*. In this style leaf cells can support different logical functions. Cell layouts must conform in height and line-pitch, but cell width (viz along the bus) can be tailored to suit the cell function. Clearly, this approach is well suited to the simplified full-custom style of leaf-cell design. Indeed, the style has found favour with designers of microprocessor datapaths.

Problems can be experienced with this approach when several different