Automatic Layout Modification

Including design reuse of the Alpha CPU in 0.13 micron SOI technology
Automatic Layout Modification

Including design reuse of the Alpha CPU in 0.13 micron SOI technology

By Michael Reinhardt
RUBICAD Corporation
Contents

Foreword .......................................................................................................................... xiii

Acknowledgments .......................................................................................................... xv

1. Introduction to IC Physical Design Reuse ................................................................. 1
   1.1 Purpose of This Book ......................................................................................... 3
   1.2 Structure of This Book ....................................................................................... 3
   1.3 History of Physical Design Reuse ....................................................................... 4

2. Physical Design Reuse in the SOC Era: Luxury or Necessity? ......................... 7
   2.1 Increasing Design Complexity ........................................................................... 8
   2.2 The Design Automation Gap and Physical Challenges of UDSM Technology .................................................................................................................. 9
   2.3 Demand for Shorter Design Time .................................................................... 11
   2.4 Early Access to the Latest Process Technology ............................................. 12
   2.5 Shortage of Qualified Engineering Resources ................................................ 13
   2.6 Design Cost: A Critical Factor in SOC Design ............................................. 14

3. Boosting Design Capabilities with Automatic Layout Modification Technology ........................................................................................................... 15
   3.1 Speeding Up Initial Layout Creation ............................................................... 15
       3.1.1 Compacting layouts .................................................................................. 18
       3.1.2 Automatic design-rule correction ............................................................. 18
       3.1.3 Device size adjustment ............................................................................ 20
   3.2 Layout Migration Methods .............................................................................. 22
       3.2.1 Re-layout of an entire circuit ................................................................... 22
3.2.2 Linear shrink method .................................................. 23
3.2.3 Flexible layout conversion using Automatic Layout Modification Technology ........................................ 25
3.3 Customizing Layout Designs ................................................ 26
  3.3.1 High-voltage or low-voltage? ........................................ 26
  3.3.2 Adjusting circuit speed .............................................. 28
  3.3.3 Low-power applications ......................................... 28
  3.3.4 Adjustments for different process options ................. 30
  3.3.5 Customizing cell libraries ....................................... 30
3.4 Achieving Timing Closure ................................................ 30
  3.4.1 The timing challenge in DSM technologies ............... 31
  3.4.2 Detecting timing problems inside a circuit ............... 34
  3.4.3 Changing transistor sizes .................................... 38
  3.4.4 Modifying wire capacitance and resistance ............. 39
  3.4.5 Clock optimization .............................................. 40
3.5 Solving Signal Integrity Problems in DSM Circuits ............ 40
  3.5.1 Reducing noise ..................................................... 40
  3.5.2 Reducing crosstalk ............................................... 42
3.6 Design Optimization .................................................... 43
  3.6.1 Timing optimization ............................................. 43
  3.6.2 Power optimization ............................................. 43
  3.6.3 Yield optimization ............................................. 44
  3.6.4 Electro-migration ................................................ 45
3.7 Optical Mask Enhancement ............................................. 46
  3.7.1 Optical proximity correction .................................. 46
  3.7.2 OPC avoidance .................................................... 48
  3.7.3 Phase shift mask correction ................................ 49

4. Characteristics and Functionalities of an Automatic Layout Modification Tool Suite ........................................ 51
  4.1 Migrating the Electrical Characteristics of a Layout .......... 51
  4.2 Signal-Specific Layout Manipulation Support ................. 55
    4.2.1 Solving signal integrity problems .......................... 55
      4.2.1.1 Signal-specific wire sizing and spacing of critical signals........................................ 56
      4.2.1.2 Shielding and moving signals to separate metal levels ........................................ 57
  4.3 Ensuring Layout Quality Throughout an Automatic Layout Modification ........................................ 59
    4.3.1 Controlling the number of notches and jogs .......... 59
    4.3.2 Handling orthogonal and non-orthogonal structures ................................................................. 60
    4.3.3 Improving connectivity ..................................... 62
4.4 Modifying Libraries Automatically ........................................ 68
  4.4.1 Processing an entire library ........................................ 69
  4.4.2 Supporting multiple grids .......................................... 69
  4.4.3 Controlling power bus width and cell height ................. 70
  4.4.4 Creating pick-up ports ............................................. 71
  4.4.5 Checking and evaluating an entire library ..................... 71

4.5 Modifying Large Databases ............................................. 72
  4.5.1 Using distributed processing ..................................... 72
  4.5.2 Modifying hierarchical layouts ................................ 73
  4.5.3 Using a hierarchy analysis and extraction tool ............. 74

4.6 Defining and Entering Design Rules ................................ 74

4.7 Preparing and Analyzing Layouts ................................... 74

4.8 Extracting Netlists ...................................................... 75

5. Integrating Automatic Layout Modification Technology and
   Physical Design Reuse into Existing Design Flows ................ 77
  5.1 The Typical ASIC Design Flow ..................................... 78
  5.2 Adding Physical Design Reuse Capability to
   the ASIC Flow .......................................................... 81
  5.3 The Layout Conversion Flow ......................................... 82
    5.3.1 The geometrical process relationship ....................... 83
    5.3.2 The electrical process relationship ......................... 84
  5.4 Applying Automatic Layout Modification Technology
   and Design Reuse Techniques to New Designs ..................... 86
  5.5 An Incremental Approach to Timing and Signal Integrity
   Design Improvements ................................................ 87
  5.6 Optimizing Power Consumption and Yield
   Using Automatic Layout Modification Techniques ............... 88

6. Applying Physical Design Reuse to Different Design Types
   with Automatic Layout Modification Technology .................. 91
  6.1 Digital Macro Blocks ................................................ 91
    6.1.1 Standard cell blocks .......................................... 92
    6.1.2 Digital full-custom designs .................................. 94
    6.1.2.1 Timing closure and Physical Design Reuse .......... 95
    6.1.2.2 Early process access and decreased ramp-up time .... 95
    6.1.2.3 Need for design portability .............................. 96
    6.1.2.4 Integrating external IP into SOC designs .............. 97
    6.1.2.5 Moving to place & route-based design flows ....... 97
6.1.3 Structured full-custom designs ............................................. 98
6.1.3.1 Datapath designs ......................................................... 98
6.1.3.2 Memory .......................................................................... 99
6.2 Analog Circuits ......................................................................... 99
6.3 Standard Cell Libraries ............................................................ 100
6.4 Physical Compilers ................................................................... 102

7. Layout Guidelines for Physical Design Reuse and
   Automatic Layout Modification ................................................... 103
   7.1 General Layout Design Guidelines .......................................... 103
       7.1.1 Laying out polygon structures ....................................... 103
       7.1.2 Positioning text .......................................................... 104
       7.1.3 Using diamond- and octagonal-shaped structures ....... 104
       7.1.4 Using 45-degree transistors ........................................... 105
       7.1.5 Maintaining one-to-one device and cell
            relationships .................................................................. 106
       7.1.6 Separating analog structures ......................................... 107
   7.2 Modifying Hierarchical Layouts ............................................... 108
       7.2.1 Fulfilling the abutting cells requirement ......................... 108
       7.2.2 Enforcing array-type base structures ............................. 110
       7.2.3 Maintaining rotated and mirrored cells ......................... 110
       7.2.4 Handling deadlock situations ....................................... 111
       7.2.5 Handling naming and text conventions ......................... 111
       7.2.6 Matching of netlist and layout hierarchies ..................... 112
   7.3 Partitioning Layouts (Floorplanning) for Physical Design
      Reuse .................................................................................. 112
       7.3.1 Reusing standard cell blocks ....................................... 113
       7.3.2 Reusing full-custom design blocks ................................. 114
       7.3.3 Reusing mixed-signal designs ...................................... 114
   7.4 Designing Layouts for Portability ............................................ 114
       7.4.1 Using orthogonal versus non-orthogonal
            structures ...................................................................... 114
       7.4.2 Using local interconnect ............................................... 115
       7.4.3 Handling butted tap contacts and shared contacts ....... 116
       7.4.4 Using buried contacts .................................................. 117
   7.5 Laying Out Analog Structures ................................................ 117

   8.1 Design Analysis Tools .......................................................... 119
       8.1.1 Electrical (parasitic) extraction tools ............................ 120
       8.1.1.1 Extraction methods .................................................. 120
       8.1.1.2 Determining critical signals ...................................... 122
       8.1.2 Critical signal extraction tools ....................................... 123
9. General Layout Modification Design Flow as Applied to the Alpha CPU Migration

9.1 General Layout Modification Design Flow

9.2 Alpha Microprocessor Project Overview

9.2.1 Migration methodology

9.3 EV7 Microprocessor Core Database Assessment

9.4 The Challenge: Layout Correction

9.4.1 Basic design-rule violations

9.4.1.1 Poly endcap

9.4.1.2 Poly spacing

9.4.1.3 Gate spacing

9.4.1.4 Local interconnect spacing

9.4.1.5 Contact spacing

9.4.1.6 Contact/local interconnect overhang

9.4.2 Generating a high-threshold voltage mask

9.4.3 Impact of hierarchy on layout correction

9.4.4 No metal modification allowed

9.4.5 Maintaining the same number of leaf cells

9.4.6 Exclusion of certain cells and “don’t-touch” areas

9.5 The Alpha Layout Modification Flow

9.6 Phase 1: Hierarchical Linear Shrink and Lambda Approach

9.7 Phase 2: Cleaning up the Database

9.7.1 Removing redundant polygons

9.7.2 Optimizing the hierarchy
9.7.3 Removing unconnected diffusion and local interconnect .................................................. 146
9.8 Phase 3: Analyzing the Database ................................................................. 146
  9.8.1 Extracting hierarchy information ......................................................... 146
  9.8.2 Merging macro blocks ................................................................. 147
9.9 Phase 4: Layout Correction Flow ............................................................. 147
  9.9.1 Preparing the layout for correction .................................................... 148
  9.9.1.1 Device extraction ............................................................ 148
  9.9.1.2 Netlist extraction ............................................................ 148
  9.9.1.3 Extracting specific layout situations ....................................... 149
  9.9.1.4 Merging polygons .............................................................. 150
  9.9.1.5 Combining polygons from different cell levels ................. 150
  9.9.1.6 Enlarging endcaps .............................................................. 151
  9.9.1.7 Enlarging end-of-line rules for LI over-contacts .......... 152
  9.9.1.8 Combining cross-shaped LI on same node .............. 152
  9.9.1.9 Creating pseudo-contacts for LI/poly and LI/diffusion ............... 152
  9.9.2 Measuring design sizes .............................................................. 153
  9.9.3 Checking and correction .............................................................. 153
  9.9.3.1 Checking the design ......................................................... 153
  9.9.3.2 Correcting the design ....................................................... 154
9.10 Phase 5: Reconstructing the Layout ..................................................... 155
  9.10.1 External verification ............................................................ 155
9.11 Project Results ....................................................................................... 155
  9.11.1 Memory and CPU time requirements ........................................ 155
  9.11.2 Amount of error reduction ....................................................... 157

10. Aspects of Memory Conversion Projects ............................................. 159
  10.1 Typical Memory Structure .............................................................. 160
  10.2 Special Requirements of Memory Conversion ........................... 161
    10.2.1 Handling memory bit cell .......................................................... 161
    10.2.2 Pitch matching leaf cells .......................................................... 162
    10.2.3 Different sizing of different memory areas .................. 164
    10.2.4 Considering all possible memory configurations .... 164
    10.2.5 Maintaining layout hierarchy for verification ............... 164
    10.2.6 Special device sizing for drivers and sense amplifiers ....................... 165
    10.2.7 Supporting a special grid for memory ports ...................... 165
    10.2.8 Typical memory conversion flow ........................................... 166
    10.2.9 Creating typical configurations ........................................ 166
    10.2.10 Reducing overall complexity .......................................... 167
    10.2.11 Analyzing memory configurations .................................... 168
10.2.12 Calculating device sizes ........................................ 168
10.2.13 Reduction or shrink factor analysis ....................... 168
10.2.14 Conversion and correction flow .......................... 169
10.2.15 Reconstructing the memory .............................. 170
10.2.16 Creating phantoms for place & route ................. 170

11. Aspects of Library Conversion Projects .......................... 171

11.1 Typical Standard Cell Structures .............................. 171
11.2 Special Standard Cell Library Conversion Requirements ...... 174
  11.2.1 Considering all possible configurations .................. 175
  11.2.2 Special device sizing of output drivers .................. 176
  11.2.3 Supporting a special grid for ports ...................... 177
11.3 Standard Cell Library Conversion Flow .................. 177
  11.3.1 Enhancing the library ................................ 177
  11.3.2 Calculating device sizing and power bus width ....... 177
  11.3.3 Adjusting the physical layout .......................... 178
  11.3.4 Verifying the physical layout .......................... 179
  11.3.5 Characterizing library cells ............................ 179
  11.3.6 Updating views for EDA tools .......................... 180
11.4 Converting I/O Cells ........................................ 180
  11.4.1 I/O cell structures .................................... 180
  11.4.2 Converting ESD and bonding pad structures .......... 181
  11.4.3 Converting the logic parts of I/O cells ............... 182
11.5 Special-Purpose Libraries .................................. 182
11.6 Gate Array Libraries ....................................... 182

Appendix 1: Key Algorithms Used for Automatic Layout

Modification Technology ............................................... 183

A1.1 Mask Operations ........................................... 184
  A1.1.1 Logical mask operations ............................... 185
  A1.1.2 Selection operations ................................. 186
  A1.1.3 Sizing operations .................................. 187
  A1.1.4 Layer operations .................................. 188
  A1.1.5 Check functions .................................. 190
  A1.1.6 Implementation options for mask operations ....... 190
A1.2 Types of Compaction Algorithms ............................ 191
  A1.2.1 Symbolic compaction– polygon compaction .......... 191
  A1.2.2 One-dimensional– two-dimensional ..................... 192
  A1.2.3 Orthogonal – non-orthogonal structures .......... 194
  A1.2.4 Flat compaction – hierarchical compaction ......... 194
A1.3 Compaction Algorithms in ALM ............................ 195
  A1.3.1 Shear line or virtual grid compaction ............... 195
  A1.3.2 Constraint graph compaction .......................... 197
A1.3.3 Iterative edge compaction................................. 198
A1.4 Compaction Algorithms for Full Hierarchical Designs,........ 201
A1.5 Limitations of Compaction........................................ 204
A1.6 Layout Correction Algorithms........................................ 205
    A1.6.1 Mask operations for design-rule correction............. 206
    A1.6.2 Grid snapping error correction after a
          linear shrink............................................. 207
    A1.6.3 Correcting hierarchical layouts......................... 207

Bibliography .................................................................... 211
Glossary .......................................................................... 215
Index .............................................................................. 219
Foreword

According to the Semiconductor Industry Association’s 1999 International Technology Roadmap for Semiconductors, by the year 2008 the integration of more than 500 million transistors will be possible on a single chip. Integrating transistors on silicon will depend increasingly on design reuse: leveraging from manufacturing capabilities and from the design effort of the past using intellectual property components from internal and external design teams.

Design reuse techniques have become the subject of books, conferences, and podium discussions over the last few years. However, most discussions focus on higher-level abstraction like RTL descriptions which can be synthesized. Design reuse is often seen as an add-on to normal design activity, or a special design task that is not an integrated part of the existing design flow. This may all be true for the ASIC world, but not for high-speed, high-performance microprocessors.

In the field of high-speed microprocessors, design reuse is an integrated part of the design flow. The method of choice in this demanding field was, and is always, physical design reuse at the layout level. In the past, the practical implementations of this method were linear shrinks and the lambda approach. With the scaling of process technology down to 0.18 micron and below, this approach lost steam and became inefficient.

When the Alpha Design Group at Compaq Computer Corporation reached the 0.13-micron technology target, we had to look for a more automated and efficient approach to physical design reuse. We tested different approaches from conventional mask manipulation, scripting, and compaction. Nearly all of them were inadequate for our task. The only viable solution was a method which is now called Automatic Layout Modification (ALM). It combines compaction, mask manipulation, and correction with
powerful control capabilities. It not only automated our partly manual approach to a high degree, it also enabled us to do tasks that were impossible before.

This book is a welcome effort to improving some of the practices in chip design today. The significance of the described methods and technology is not only in the area of making physical design reuse much simpler and more efficient, but also in new applications and fields which are very relevant to the implementation of ultra deep sub-micron (UDSM) circuits. These specific areas are:

- Timing closure
- Solving signal integrity problems
- Yield optimization
- Low-power design
- Design for manufacturing: optimal proximity correction (OPC) and phase shift mask (PSM)

This book is a comprehensive reference work on Automatic Layout Modification which will be valuable to VLSI courses at universities, and to CAD and circuit engineers and engineering managers.

Goran Zelic
Principal Software Engineer
Alpha Development Group
Compaq Computer Corporation
Shrewsbury, Massachusetts
I would like to thank the uncounted number of people who supported and contributed to this work.

Especially I want to thank the following people for their contributions, reviews and suggestions: Gabriele Eckert, Juzer Fatehi, Mike Keating, Rick Kelly, Jim Lipman, Resve Saleh, Michael Santarini, Josef Stockinger, and John Yarborough.

A very special thanks goes to Ann Thryft for her patience and persistence in editing the book.

I thank Will Sherwood for the contribution of the cover picture and the photo of the Alpha chip.

My special thanks to Goran Zelic and his team for the contributions in Chapter 10, which contains a lot of data from the Alpha CPU conversion project done at Compaq Computer Corporation.

I also want to thank everyone else who helped in guiding this book toward publication.

Michael Reinhardt
Rubicad Corporation
San Jose, California
Chapter 1

Introduction to IC Physical Design Reuse

In general, design reuse is the ability to recycle all or part of a design by using it again in a different application. The effect is similar to moving from a typewriter to an electronic word processor. The savings in time and effort can be enormous when reusing existing text for a new purpose without having to re-type it all over again.

The assumption in IC physical design reuse is that similar savings can be achieved in electronic design. In electronic design, design reuse using software descriptions of circuits, or soft IP (intellectual property), is distinguished from the reuse of the final physical implementation of a circuit, called physical design reuse. Therefore, design reuse, and specifically physical design reuse, has been proclaimed as the solution to closing the design productivity gap, which is the gap between the production capabilities of semiconductor fabs and the ability to design complex semiconductors. The basic idea is that functional macro blocks will be designed once and then combined into more complex systems on a higher level. This is similar to the way printed circuit board designers use pre-designed ICs to design a board; they do not design each IC first. Because the main design effort is in the design of the functional blocks, reusing the blocks in multiple designs will result in shortening the design process and making it more cost effective. However, there have been, and still are, several barriers to overcome before the design reuse of macros becomes a widespread practice throughout the semiconductor industry.
One major barrier is the fact that the designer, division or company which originally created a particular design is not usually the one which can benefit from the reuse of the design. Even so, it is clear that the company, the industry, and society as a whole will ultimately benefit from effective design reuse because, among other things, it can dramatically reduce the cost of electronic systems. To overcome this particular barrier, companies tried to create incentives for design reuse, new business units were founded, and the business model of chip-less semiconductor companies was invented. Chip-less semiconductor companies design IP modules and license them to other companies. Those companies combine the modules with their own or with third-party blocks into new silicon systems. The most successful chip-less companies provide processor cores, memory generators, analog blocks, and libraries. However, the number of companies based on this business model is still small.

Another barrier to introducing design reuse is the missing infrastructure of electronic design automation (EDA) tools. Most approaches for design reuse look more like a re-implementation than a re-use of the final design. The other major barrier is an automation gap in the current design flow between the physical implementation using place & route and the consideration of advanced physical effects on the wafer. These effects like timing and signal integrity can be simulated, but they can’t be automatically corrected or adjusted in the current design flow. However, in the long run, we have to find a solution for the automation gap and an efficient design reuse methodology to keep the semiconductor industry--and specifically, the fabless semiconductor sector--on a growth track, which will benefit the EDA and the semiconductor industry as well. Otherwise, the challenges of designing complex circuits may limit the size of the electronics industry as a whole and with it, the size of the EDA market.

The third barrier, and often the most difficult to overcome, is a psychological one. Engineers and designers are often reluctant to reuse their colleagues’ work, especially when they do not know the particular colleague, nor the work, in detail. This is true not only for IC designers, but also for software engineers, and other engineers. The reason is not that most engineers are egocentric, but that the risks involved in reusing existing blocks are very high. Normally, there is no guarantee from IP providers that their IP will work in the final implementation. Often, license agreements prevent those who buy IP from changing it and adapting it for other purposes. But the most frequently-cited reason is that the reuse effort is close to, or as high as, the effort involved in implementing the circuit from scratch.

The situation is beginning to change. One major factor is the downturn in the cyclical semiconductor industry: in a tough market situation, companies have to work more efficiently. They recognize that putting more people on a
design will not reduce time to market or cost to market. In addition, the shortage of experienced people in design implementation demands a higher degree of automation.

Huge investments were made to develop tools that can implement VHDL down to GDSII, while automatically considering all implementation problems, such as timing closure, crosstalk, noise, and power consumption. But until now, these new tools have not shown the promised results in overcoming the design gap.

1.1 Purpose of This Book

This book focuses on physical design reuse (PDR) and the techniques of implementing it, called automatic layout modification (ALM). Physical design reuse is one of the oldest methods of shifting designs from one process technology to the next and from one application to another. Aside from this application, the underlying technology of ALM can be used for many others, which are related primarily to the problems of ultra deep sub-micron (UDSM) technologies and designs. The main UDSM problems are timing closure, noise, power optimization, optical proximity correction, and phase shift mask implementation. The interesting point is that the techniques for physical design reuse can also be applied to designing new circuits while shortening their design cycles, even when no physical design reuse is involved.

The target audience for this book includes engineering managers, design engineers, technology development engineers, device and product engineers and CAD engineers. The book provides a general overview of physical design reuse methodology, along with its applications, design flows, and features. It also gives users practical guidelines on how to execute a physical design reuse project by discussing in detail three specific types of applications: libraries, memory, and microprocessors. These applications are based on real projects conducted by Rubicad Corporation or its customers using the LADEE tool suite. The microprocessor project describes the conversion of the Alpha CPU of Compaq Computer Corporation.

1.2 Structure of This Book

The main goal of this book is to familiarize the reader with physical design reuse techniques and how they can be applied to many actual design problems, including those that occur in new designs. The book begins with a discussion of the motivations for design reuse in Chapter 2. Physical design reuse means that the physical design has to be modified. The core principles of doing this are associated with compaction and layout manipulation.
software. Chapter 3 discusses in detail the potential applications for ALM technology, which can help boost design capabilities in physical design reuse applications, as well as in many applications suited to new designs. Chapter 4 defines the characteristics and functions of an ALM software tool suite. How ALM technology can be integrated into existing design flows is discussed in Chapter 5, and Chapter 6 covers physical design reuse for different design styles. Guidelines for layout style are given in Chapter 7 and Chapter 8 discusses other design tools which may be needed for a complete design reuse flow. Chapters 9 to 11 discuss the challenges designers may face in the execution of a physical design reuse project. These chapters summarize the experience of real projects in standard cell libraries, memory and memory compilers, and large macro blocks or full-custom designs. The largest design is Compaq Computer Corporation’s Alpha CPU, which is discussed in detail in Chapter 9. Chapters 10 and 11 discuss memory, memory compiler, and standard cell library projects.

The appendix contains a chapter about algorithms for physical design reuse. It deals with physical mask operations and several compaction algorithms in more detail.

1.3 History of Physical Design Reuse

Design reuse in the semiconductor industry is not a recent invention. The use of TTL logic chips on the board level, which is a type of design reuse, was duplicated by the development and use of standard cell libraries. The move to standard cell libraries was not a natural one. Many designers were designing a new library with similar functionality for each specific chip. However, the semiconductor industry had one major disadvantage compared to other industries in the field of design reuse. This is the rapid change in process technologies, which makes the use of the same library for different processes difficult or impossible.

To overcome that disadvantage, a lot of research was done in symbolic layout and compaction during the 1970s and 1980s. However, this approach was not successful in industrial applications, because the required computer performance for such approaches was not available at that time and because the linear shrink approach was much more compelling.

The linear shrink approach is the easiest way to convert a physical layout design to a new or different process. It can be applied as a soft shrink on digital data like library cells, or as an optical linear shrink using optical equipment during the mask making process. An optical linear shrink is applied to the whole chip; its advantage is that any arbitrary shrink factor can be used. A soft shrink is less flexible because the resulting layout has to
be design-rule correct and all structures must be adjusted to the drawing grid.

With the latest UDSM technology, the optical shrink is losing steam, because designs can no longer be shrunk linearly. Different structures have to be sized differently: some structures become larger, while others become smaller. The reasons for this are optical and electrical constraints, such as the wavelength of available light sources, wire resistance, and heat dissipation. Many of these problems can be eventually solved by adapting layout compaction technology to the new needs of sophisticated layout manipulation, in order to solve new problems like optical proximity correction (OPC) and phase shift mask (PSM) conflicts. This automatic layout modification technology can be used for the design reuse of physical designs, as well as for solving many design problems in new designs.
There have been many discussions in industry and academia about whether design reuse is possible or makes any sense at all. Some groups are proposing that a new type of specialized designer is needed, who only designs hard or soft IP for reuse. Some say that design reuse is basically impossible to do in an efficient way. This chapter examines different aspects of the design process and the electronics industry. The bottom-line question is whether the electronics industry has a choice about implementing design reuse: is it only a luxury, or is it becoming a necessity?

As stated in Chapter 1, design reuse is not a new invention. Physical design reuse has been common practice for the production of complex designs, such as microprocessors or microcontrollers, for a long time. During the last few years, many companies have promoted the design reuse approach in the form of re-synthesizing soft IP. The reasons are obvious: soft IP fits into the existing design flow and is easily accepted by designers. The tools for using it are already installed and only gradual performance improvements are required to increase the output. However, although this approach appears to be straightforward and agreeable, it does not provide the design productivity necessary for creating large system-on-chip (SOC) designs at the required cost level and within the design time required to comply with today’s short market windows.

This chapter will discuss the main reasons why physical design reuse is becoming a necessity in the SOC era: increasing design complexity, the
design automation gap, the physical challenges of UDSM design, the
demand for shorter design time, the need for early access to the latest
process technology, the shortage of qualified engineering resources and
design cost issues.

2.1 Increasing Design Complexity

When people talk about design reuse, the term “design complexity” is
often used. However, this means different things to different people.

The first question to be answered is by what metric design complexity is
being measured. The possibilities include the number of gates or
components, area size, size of the design team, and the number of different
disciplines and types of expertise involved in a project. The measurement of
design complexity should include a metric for the costs of a design project.

In addition to the factors which can be quantified, there are other factors
which can be estimated or quantified only with difficulty, if at all. Examples
of these include the effect of clock frequency on the design effort, stability
of the process technology, and communication with system integrators.

Suppose that design complexity is being measured by the number of
transistors or gates, or by area. If the real effort of producing or designing a
circuit is examined, these figures become less relevant. Design complexity
has grown so much because of circuit scaling; for example, the increase in
capacity of memory chips from 256 KBytes to 256 MBytes. The difference
in design effort between these two memory densities is not all that great,
since there are many repetitive elements on a memory chip. The design time
for 256 KBytes or 256 MBytes is almost the same.

A similar situation is encountered with design structures like datapath
designs, which have increased from 16 bits to 64 bits. In this case, the
existing structure is simply multiplied. The absolute time to design these two
structures is more or less the same when the circuit architecture is not
changed.

However, in SOC design, there are many different components on one
chip. These include a microprocessor, several different types of memory, an
execution unit, a DSP, some random logic, and lots of dedicated components
for connecting the circuit to the outside world, such as A/D and D/A
converters, Serial Communication Interface (SCI), Synchronous Periphery
Interface (SPI) and digital bus interfaces. Aside from these, there is analog
functionality, which requires a major design effort.

Therefore, the effort required for SOC designs cannot be compared by
using the number of transistors. An SOC design of 10 million transistors can
be much more complex than a memory of 100 million or a CPU of 20
million transistors, because the repetition factor in an SOC design is
naturally lower than in a conventional design. It also becomes clear that it is not enough merely to have faster tools or faster methods. Designing an SOC from scratch would require much larger design teams simply because of the many different components, which often require different kinds of expert knowledge. So the communication requirements would grow exponentially, which would undoubtedly cause project delays. Unfortunately, this problem cannot be easily solved by new means of communication, such as the Internet or remote conferencing technology, because each of them provides only a new means of information transfer, and not a new means of information processing and organization.

Design reuse is a way to geographically decentralize the design effort and spreads the design effort over time without increasing the communication effort.

2.2 The Design Automation Gap and Physical Challenges of UDSM Technology

The current ASIC or IC design flow has evolved in a way similar to the software design flow. Some elements of the IC design flow were even modelled after the software flow. The software design flow has evolved from programming in a machine language that is CPU-or computer-specific, to programming in a high-level abstract programming language. Code written in languages such as C++ or Java can be automatically translated to different target computers running different operating systems. There is no need to simulate the final object code every time a translation is done in order to verify the work of the compilers that do the translations. Software designers have a nearly 100% trust in their compilers.

At the beginning of the 1980s, academia and industry started to develop a similar approach to IC design. The high-level programming languages were called hardware description languages (HDLs). The most popular HDL today is VHDL. In IC design, the compilation of HDL code into the register-transfer level (RTL) and gate level netlist is called synthesis. This step can be compared to the first phase of a compiler that translates high-level code into low-level meta-languages, which are still not CPU-specific. Place & route in IC design is the equivalent of the final task of CPU-specific code generation and linking in software design. P&R creates the process-specific implementation of the IC. The idea was that synthesis and P&R are fully automatic tasks that do not require extensive verification at the physical level. This was true for process technologies down to about 0.5 micron (see Figure 2-1).
As process development moved below the 0.35-micron technology node, the capability evolved of integrating a real SOC containing many millions of transistors. However, in these new process technologies designers have to deal with new design challenges, which previously they had not needed to consider. These challenges also increase the time needed to design traditional circuits in these new technologies.

The specific design challenges are: timing closure problems caused by higher wire delays compared to transistor delays; signal integrity problems; power problems; a larger number of critical features that may reduce yield; and new mask requirements like optical proximity correction and phase shift masks. All of these problems require a greater understanding of and focus on physical design, which is not supported by today’s ASIC design flow that combines logic synthesis and automatic place & route. Because of this, a lot of verification is required to check whether the final layout fulfils the requirements of the initial VHDL specification and the process technology. These feedback loops require a lot of iterations and manual analysis and changes in the design. This is the automation gap in UDSM processes (see Figure 2-2).

Inserting automated layout modification techniques into the design flow can diminish the automation gap in IC design. Physical design reuse can prevent multiple physical re-implementations of a circuit.
Thus, design reuse in the SOC era is not a “want” but a “must.” The only question is whether the design must be reused at a lower level of abstraction, or if there is an automatic design flow that can guarantee a flawless design synthesized from a VHDL specification. Such an automatic design flow will, of course, require a method of modifying, manipulating, and automatically and intelligently adjusting physical designs. That method is called automatic layout modification.

2.3 Demand for Shorter Design Time

In the early days, nearly all electronic systems were board designs. Boards have a design cycle of 4 to 6 weeks and are relatively easy to modify. With the move to integrated circuits and application-specific ICs (ASICs), part of the board design became IC design. However, the design cycle for an ASIC today is still 4 to 9 months in most cases, which is four to six times longer than the board design cycle. The pressure to shorten the IC design cycle originates less from shorter consumer or end product cycles than from the fact that IC design is considered as a substitute for board design.
SOC designs are far more complex than early ASIC designs. A single SOC is substituted for a whole board or even the complete system. Most experts agree that this can only be achieved by using a design reuse-based design flow for SOCs. Otherwise, the requirements of system designers can never be fulfilled.

That is because each SOC chip is basically an ASIC made for a very specific application. In the past, electronic systems were assembled using a number of these pre-produced circuits that were developed over a period of years by different suppliers and manufacturers. Now, an SOC ASIC is created for a single specific application: one chip comprises the entire electronics of a TV, or a radio, or a car’s control system, for example. The market for this type of chip design, which is mainly a consumer electronics market, is changing very fast and product life cycles can be only three to six months long. Yet, to design an SOC chip of 10 million transistors or more from scratch can take several years.

Another factor contributing to today’s design challenges is that time to volume has decreased significantly. It took about 12 years to sell 1 million color TVs and three years to sell 1 million cellular phones, but only about one week recently to sell 1 million units of the latest video game console. This means not only that the design cycle and time to market must be reduced, but also that a design must be right the first time.

These changes in market requirements are not new: other industries have seen them before. A good example is the automobile industry. The development cycles for cars has been shortened and the variety of models increased by introducing so-called “platform design”. In platform design, different models are created by selecting from among the same set of components. Often, only the bodywork is different and the main mechanical parts under the hood, such as engine and braking systems, are the same. Some people propose a similar approach in the electronic industry by introducing huge IC platforms which can be customized for different applications by software alone.

Today’s design technology does not fulfil the requirements for current and future market conditions of complex SOC designs. Therefore, design reuse is required to assemble complex SOC chips faster from pre-defined circuits. The entire industry agrees on this conclusion. This book will discuss the methods and tools needed to accomplish this in the most efficient way.

### 2.4 Early Access to the Latest Process Technology

The search for creating more applications and new markets for IC designs has created more and more complex systems. The design time for these very large systems, containing 10 to 100 million transistors or more,
has become longer and longer. In addition, these systems always require the latest process technology in order to be competitive. Therefore, process designers have been pressured to provide access to advanced, but still uncertain, technology information like design rules and process parameters. The effect is that designers often have to deal with processes that are in flux. Since there are more process revisions than before, so-called in-between or intermediate process steps have been introduced.

Because an SOC design and a process technology are most often developed in parallel, designers have to deal with unstable process parameters. New SOC designs are almost always implemented with process technologies that are in production for less than a year and whose yields are not as high as older established process technologies.

Therefore, the industry needs a method for dealing with designs with uncertain process parameters. One solution would be to apply automatic layout modification technology to adjust the final layout design according to the design rule and process parameter changes. Another would be to use a fully automatic design flow on the way down from the VHDL level.

\section{2.5 Shortage of Qualified Engineering Resources}

The most valuable resource of any design house or IC manufacturer is the expertise of its people. This expertise cannot be easily transferred from one person to another, nor can it be easily documented or provided in some other way.

Engineering management is challenged with fluctuations and shortages in electronic engineers, especially in economic boom times. This is particularly true for complex design projects that last several years and require a number of large teams. Design reuse techniques may not eliminate this problem, but they can reduce it significantly. The most valuable techniques would produce a method that reuses the objects that contain the most design value. In this context, physical layout is more valuable than the VHDL code, provided that both representations can be reused in a design flow. The reason is that all the synthesis effort already performed by an engineer is stored in the hard IP, represented by the physical layout design.

Using physical IP blocks in the way that ICs are used on a printed circuit board can significantly reduce both design complexity and necessary communication between designers. In this way, circuits which were designed independently of each other or by different groups can be combined quickly, so that the fluctuation and/or shortage of engineers can be overcome because the knowledge is stored in the designed block. When the pre-designed blocks are of medium complexity and can be designed in a short time span, this is an ideal way to store and distribute engineering
knowledge. In this way, project teams are less vulnerable to the fluctuation of key personnel.

2.6 Design Cost: A Critical Factor in SOC Design

Since an SOC represents the entire electronic system of a certain application—such as a cellular phone, or a TV, or even the particular model of a certain vendor—it is much more specific than is a standard product like memory. Therefore, there are far fewer potential applications for a particular SOC, which results in a much lower production volume for most SOCs compared to standard products, and a much higher market risk as well. The volume of an SOC design will fall even farther with increased integration of different functions; but this represents a growing conflict with increasing design costs. When the non-recurring engineering (NRE) costs cannot be shared among several designs, the unit cost increases and will limit SOC designs to very high-volume applications.

This means that companies will focus much more on design costs in the SOC era. Design flow and design knowhow will grow in importance relative to the importance of the production technology, which dominated decisions in the era of standard products.

A substantial part of an SOC’s functionality is coded in software. The software portion of an SOC application is significantly greater than that of a memory or microcontroller design, sometimes several megabytes of binary code. Productivity in software development can be measured by the amount of tested code written by a software engineer. In terms of code length, this productivity has not increased much over the years. The increases in productivity that have occurred have been achieved mainly by increasing the level of abstraction. Increasing the level of abstraction in software design means to increase the use of more complex, pre-defined library routines.

With growing SOC complexity, the development testing of embedded software can become an even larger problem than testing the hardware. Proposals to implement an increasing amount of functionality in software instead of hardware may not lead to faster and cheaper SOC development. Specifically, the software quality in these complex systems which are used in critical applications are a major area of concern.

To enable SOC designs for low- and medium-volume products, the NRE costs of these designs must be shared across different product lines. This is only possible with the modular development of independent components that will be combined at the latest possible design stage.
Chapter 3

Boosting Design Capabilities with Automatic Layout Modification Technology

Automatic layout modification (ALM) technology can provide the solution for many design challenges in the era of ultra deep sub-micron technologies. However, until now only a few automatic layout techniques have been widely used: automatic place & route, and mask layer operations. This chapter will introduce the applications for an intelligent layout modification approach that can boost the design capabilities of today’s engineering teams by speeding up initial layout creation, migrating layouts, customizing layout designs, achieving timing closure, solving signal integrity problems, optimizing designs, and developing test chips for new technologies.

3.1 Speeding Up Initial Layout Creation

The first application area for ALM technology is boosting the speed of initial layout designs. The current approaches used to create layouts automatically are automatic place & route (AP&R), physical layout compilers, and symbolic layout and compaction. However, most layouts created for memory, high-speed microprocessors and microcontrollers, and analog designs are still drawn manually. All of these methods can be significantly enhanced by using ALM technology.

The AP&R approach can be enhanced by improving the layout after the initial place & route is done. These improvements can include implementing engineering change orders (ECOs) to improve timing, power and signal
Chapter 3

integrity; inserting redundant via cuts to improve yield and performance; or automatically fixing remaining design-rule errors.

In the past, several fully-automatic layout systems, also called module generators, have been implemented. Most of them are designed for a very specific type of circuit, such as memory compilers or datapath compilers. Because of the fast progress in process technology and the high maintenance cost of these systems, many of them were abandoned. The basic problem with module generators is their long development cycle, which is in conflict with the shorter technology cycles and time-to-market requirements of today’s IC designs. In order to provide efficient module generators in the future, technology-independent compilers are needed with much longer lifetimes than today’s technology-specific compilers. ALM technology provides the basic technology for more flexible technology-independent module generators. It can increase the lifespan of these generators and ensure that they are available shortly after the process parameters and design rules are known and stable.

![Contact in 0.5-micron technology](image)

![Contact in 0.18-micron technology](image)

Figure 3.1. Comparison of grid size and feature size in 0.5-micron and 0.18-micron technology.

One ever-increasing bottleneck in layout design is the creation of full-custom layouts. Full-custom layout design includes the standard cell libraries
needed for AP&R, as well as leaf cells for module generators, memory design, and custom analog and digital designs. The amount of effort required for full-custom layout is increasing because of larger, more complex circuits, smaller layout design grids, and more complex design rules. Figure 3-1 shows a metal structure in 0.5- and 0.18-micron technology using a grid of 0.05 micron and 0.01 micron, respectively. The grid has been changed by a factor of 5 but the structure size has changed only by a factor of 2.77. This difference makes it harder to edit such a layout, because in order to draw the layout accurately with minimum design rules the layout designer has to zoom in much farther. Thus, even a small standard cell can become a large cell. More complex and global design rules present even greater challenges.

In today’s UDSM processes, strict density, antenna, and wire length rules are standard. However, these are global rules, meaning the designer cannot decide at the cell level whether or not the rule may be violated. Figure 3-2 shows examples of antenna and wire length rule violations. The initial cells A, B, C and D are design-rule correct, but the combination of those four may create errors.

![Figure 3-2. Design-rule correct individual cells may, when combined, cause violations of global design rules.]

The initial route taken for overcoming the bottleneck in layout design was symbolic layout and compaction. However, this approach was not a big success because of the limitations at that time of computation power and compaction tools. The next route was to avoid full-custom layout as much as possible by maximizing the use of AP&R. But AP&R also has limitations. It cannot be applied to all circuits, specifically not to memory, analog, and high-speed designs. Today, an increasing number of circuits have to be implemented in full-custom style because of speed requirements. ALM technology can help speed up full-custom design by allowing the designer to
draw the layout with a larger drawing grid, or even violate design rules and let the errors be fixed automatically.

The subjects discussed in the following three sections--compaction, automatic correction of design-rule violations, and device sizing--can help significantly increase the performance of manual layout designs.

3.1.1 Compacting layouts

Layout compaction is the process of reducing the area of a layout without compromising circuit performance or design-rule correctness. The basic idea is to create a rough layout, either manually or automatically, and then compress it by compacting the layout structures for final use. Compaction can be used standalone, or as part of a schematic-to-layout synthesis process.

Compaction can be used for both digital and analog circuits. Layout compaction can be combined with transistor sizing and wire sizing to assure proper circuit functionality. The core algorithms of compaction can also be applied to other applications of ALM.

Different compaction algorithms are discussed in the Appendix.

3.1.2 Automatic design-rule correction

One of the most common tasks in design verification is performing a design-rule check (DRC). Modern deep sub-micron processes below 0.18 micron often contain complex design rules which are not easy to consider during initial drawing. DRC is an iterative process of checking and correction that can demand a high percentage of layout design time, especially in analog and full-custom designs such as memory. An automatic design rule error correction process can save significant design time.

In full-custom designs, cells are used in different places and often overlap each other. The problem with manual design rule correction is that all instances, or placements, of the cell have to be considered simultaneously. With a cell that is placed in many different contexts, this is a time-consuming task. A hierarchical design-rule correction tool can correct the cell in all placements without creating additional violations. It corrects the cell for all instances simultaneously without the need for creating additional copies of the cell for different places or instances.

Layout productivity in UDSM technologies has suffered from the smaller technology grid. Grid sizes of 5 or 10 nm make even small cells virtually large, because designers have to zoom much farther to position the polygons exactly. In addition, design-rule correction tools do not require polygons to be on the exact grid. Designers can use a larger grid and ALM technology to adjust the layout.
Compaction has been used for this task, but doing so produced the side effect of moving around things that are already correct and should not be touched at all. ALM technology can be used to implement automatic design-rule error correction. With this technology, all errors or only specific errors will be corrected and no other structures will be touched. This is of great importance for analog circuits because the electrical circuit parameters will be modified only at a minimum. Additional parasitic capacitance or mismatches between matched structures can be prevented. Figure 3-3 shows the difference between correction and compaction.

The typical physical design reuse approach is a linear shrink. Often, the degree of shrink can be extended when small layout modifications are made to prevent design rule violations. These adjustments are usually done with mask layer operations. Mask layer operations are very limited because they do not allow the movement of polygons and they operate on a cell level without having knowledge of the structures on top of and outside the cell. Not knowing the structures on top and outside of the cell limits the number of design-rule violations that can be corrected and often demands the flattening of layout structures. The flattening of cells will increase the data amount, which in many cases is not acceptable.

Automatic design-rule correction is even applicable to AP&R. Because design rules have become more complex some of them – like antenna rules,
wire length rules, and area rules – cannot be easily integrated and implemented with existing AP&R tools. Often, implementing these rules will result in non-optimum results. Applying automatic design-rule correction to layouts created by AP&R tools can minimize the problem.

3.2.3 Device size adjustment

A typical design flow consists of defining a VHDL description or a schematic; synthesizing and simulating it; and doing the layout either automatically with place & route or in full-custom style with the layout editor. This flow makes the assumption that wire capacitance and resistance need not be considered. However, this is not true for critical analog circuits or for high-speed digital circuits, and the problem becomes more critical as the technology becomes more advanced and circuit size increases.

This is because advanced technologies have smaller feature sizes and the relation of wire delay to transistor delay is increased. (For more details on the timing problem in deep sub-micron see section 3.4.) Thus, all critical digital and analog circuits must be simulated after a layout has been created. For this purpose, the passive elements like wire capacitance and resistors must be extracted with their actual values. The netlist has to be extended and then simulated again. Today, this process is part of the normal verification flow. Now the question is, what happens when the result does not meet the specification? In this case, the schematic or VHDL description has to be adjusted to compensate for the parasitic elements inside the layout, and this new schematic must be implemented in the layout again. Often, this process requires many iterations of the synthesis, P&R, extraction and timing analysis loop. Because this is very time consuming and can cause unexpected product delays, this process should be shortened. Extraction and simulation, as well as place & route, are fairly automated already. Delays occur mainly in analyzing the timing results and implementing the changes by re-synthesizing or re-routing the design.

Automatic layout modification technology can solve this problem by implementing the new device sizes into the layout, as shown in Figure 3-4. This is not limited to simple N and P transistors; resistors and capacitors can also be sized in the same way. The approach is general, in that it can be applied to full-custom and ASIC-type design styles. The modification should also include modifying shapes, such as adding or reducing the number of fingers, modifying meander-shaped transistors, and modifying devices that can have an arbitrary shape, like capacitors.
ALM technology can also be used to implement ECOs, because additional devices can be squeezed into the layout even when design rules are violated, as shown in Figure 3-5. The de-compaction capability of ALM technology ensures a design rule-correct result.

*Figure 3-4. ALM technology can reduce the number of iterations usually needed for analyzing timing results and implementing changes by implementing the new device sizes into the layout.*
3.2 Layout Migration Methods

The traditional application of compaction technology has been technology-independent layout design. The migration of a layout to a new or different fabrication process is therefore one of the main applications of ALM technology. This section discusses traditional approaches to layout migration, including re-layout of the circuit and the linear shrink approach, as well as the application of ALM technology to layout migration.

3.2.1 Re-layout of an entire circuit

One of the traditional methods of migrating to a new fabrication process is simply to re-layout the whole circuit. This approach, which is very common, is extremely labor-intensive when applied to full-custom circuits, such as microprocessors, memory, and large analog designs. It is economical if automatic layout design is used. The re-layout approach for full-custom designs has been applied in areas where costs are not the major concern, specifically in high-volume designs such as microprocessors and other high-performance devices produced by the big integrated device manufactures (IDMs).

The re-layout process provides the most freedom and allows designers to use the technology in the most optimum manner. It is often combined with enhancements of the existing circuit and also allows optimization at the circuit level. In short, the entire design cycle is repeated. Nevertheless, this process also carries risks similar to those of a new design, especially when the job is carried out by a new team. Because of the demand for early access to information on the latest process, final design rules are not available at the start of a project, so many parts of a design may need to be redesigned several times.

<table>
<thead>
<tr>
<th>Layout Type</th>
<th>Number of Transistors</th>
<th>Number of Signals</th>
<th>Time to Handcraft (hrs.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard Cell</td>
<td>2</td>
<td>2</td>
<td>2 - 8</td>
</tr>
<tr>
<td>Standard Cell</td>
<td>16</td>
<td>6</td>
<td>8 - 16</td>
</tr>
<tr>
<td>Datapath Cell</td>
<td>2</td>
<td>2</td>
<td>4 - 8</td>
</tr>
<tr>
<td>Datapath Cell</td>
<td>16</td>
<td>6</td>
<td>8 - 24</td>
</tr>
<tr>
<td>Random Logic Cell</td>
<td>2</td>
<td>2</td>
<td>1 - 2</td>
</tr>
<tr>
<td>Random Logic Cell</td>
<td>16</td>
<td>6</td>
<td>4 - 12</td>
</tr>
</tbody>
</table>

*Figure 3-6. Design times for manual layout design.*
Layout design times are difficult to estimate, because the performance of individual designers varies widely, and the design time depends on special requirements like minimum area or minimum capacitance. However, Figure 3-6 shows some typical design times of different layout types, which can be used as the basis for an estimate.

Today, the re-layout approach is limited by economical and management factors, including shrinking market cycles that make time to market a critical factor, the availability of enough experienced personnel for the tasks, and high fluctuations of engineering personnel within the industry.

3.2.2 Linear shrink method

The linear shrink approach is the most widely used for porting a design to a new or different technology. With a linear shrink all layout structures will be reduced linearly by the same factor. This is either done at the mask level or at the optical level during processing. The optical approach has the advantage that it need not care about the layout design grid. The linear shrink approach is based on the fact that CMOS technology is scalable for many digital applications and process families. However, there are always structures in a design that do not scale. These are I/O structures such as pad windows, and analog structures such as capacitors and resistors. Designers increase the size of these structures before applying the linear shrink so that the size after the shrink is correct. This is usually done manually and is often very time consuming.

The linear shrink approach does not change the relative sizes of the different devices. For example, the ratio between N and P transistors will stay the same after a linear shrink. This is both the strength and the weakness of the method. It is a strength because the method does not create unexpected different electrical characteristics. It is a weakness because designers cannot change the electrical characteristics when necessary. The linear shrink approach is basically not a design technique, but a characteristic of processes. A design can be shrunk from process A to process B, but a design is not shrinkable, per se. Large IDM companies take care to ensure that their old and new processes are compatible, so that they will be able to shrink their older designs to the new technology. They often plan these shrinks ahead of time by using design rules in the current process that are larger than necessary for critical structures such as resistors and capacitors.

One necessary condition for a linear shrink is that design rules change linearly. If the first metal layer pitch is reduced by 20% and the second metal layer pitch is not changed at all, a linear shrink is not possible. These differences between the shrink factors of different design rules can be
compensated for partially with mask operations or by manual design rule adjustments. However, the effort of manually adjusting the layout can be huge in UDSM processes, since designs are large and many design rules cannot be reduced to the same scale because of electrical and process limits. Examples are the poly endcap rule for transistors and the nwell spacing. It is even possible that some design rules will increase their values in a smaller process geometry.

The automatic adjustment of design rules using mask operations is done by oversizing and undersizing operations, as well as by using logical mask operations. As this process is executed on a cell-by-cell basis, it often creates errors at the cell borders. The use of undersizing operations often causes opens at the cell borders when cells abut, as shown in Figure 3-7. To prevent them, the hierarchy has to be exploded, which is very undesirable because it increases the already large layout database. The correction of design-rule violations after a linear shrink is one area where ALM technology can be of great service.

![Figure 3-7. The use of undersizing operations often causes opens at cell borders when cells abut.](image)

Another problem with soft shrink is the adjustment of the drawing grid. A soft shrink is done on the layout database, which requires the resulting layout to be compliant with the correct drawing grid resolution. This is usually achieved by grid snapping, but in a hierarchical layout structure grid snapping can cause opens and misalignments in the database (see Figure 3-8). These opens cause design-rule violations. They can either be corrected manually or adjusted automatically using ALM technology.
3.2.3 Flexible layout conversion using Automatic Layout Modification technology

Automatic layout conversion is an approach that overcomes the limitations of the linear shrink. This approach is based on compaction technology but includes some additional properties. Compaction technology considers the geometrical design rules. So does conversion technology, but in addition it considers the electrical constraints and parameters. For example, conversion technology is able to do meaningful transistor and wire sizing, as well as maintaining other non-minimum structures. Most approaches will not modify the routing or the topology of the circuit.

The input for automatic layout conversion software is the layout design, the design rules, device sizing information and instructions for the conversion (see Figure 3-9). The layout is usually provided in GDSII or a similar layout format, or via an application programming interface (API). The design rules are defined in EDIF or in a proprietary format. The conversion parameters control the general degree of the compaction, or shrink, and the structural changes. These changes are the insertion or deletion of jogs in the layout, as well as changing 45-degree structures into 90-degree structures and vice versa.

![Figure 3-8. In a hierarchical layout, grid snapping can cause misalignments.](image)

![Figure 3-9. Automatic layout conversion flow.](image)
Electrical changes can be considered by defining the sizing of electrical devices, for example, transistors, capacitors, resistors, and diodes. Device sizing can be used to change all devices with a single function or to change specific devices to a certain size. In this way, the N to P relation between transistors and the capacitor value can be adjusted in the whole layout.

Sizing capabilities can also be applied to parasitic devices that are not documented in the schematic, such as wire resistance and capacitance, source and drain resistance of transistors, and contact resistance. These capabilities help to adjust the number of contacts in the source and drain area of transistors, as well as control the width of power lines.

Layout conversion technology is most often applied to macro blocks. It is normally used for those applications where a linear shrink does not work, for example, when the process shift goes beyond a fab line from one vendor to another. Layout conversion is also used to combine circuits that are designed in different processes on a single piece of silicon. Layout conversion is the method of choice in this case, because the different source processes often have different electrical characteristics which need to be adjusted.

### 3.3 Customizing Layout Designs

Customizing a layout design is the process of adjusting a design to different target applications. Each design is usually made for a specific market segment—such as an automotive, mobile, or telecommunications application—and may later be applied to a different market segment. Often, a total redesign of the circuit is too risky, too time consuming, or too expensive. This section discusses how automatic layout modification technology can be used to perform the necessary changes.

#### 3.3.1 High-voltage or low-voltage?

The decision of which supply voltage level to use for a certain device depends on several factors: general system supply voltage, availability of complementary design parts at the same voltage level, prices of other system parts, and different markets. For example, in the automotive industry a 12-V supply voltage is common, and for mobile applications 2.5 V is normal. Many circuits and functions can be used in a variety of applications. Examples are microcontrollers, DSPs, A/D and D/A converters, and all types of memory. ALM technology can be used to convert a circuit to a different supply voltage.

A very typical modification is changing the gate length. A transistor designed in 0.25-micron technology can usually support a supply voltage level of around 2.5 V. If the voltage level increases it will break and create
permanent damage to the circuit. To operate a 0.25-micron design at higher voltage levels, the gate length has to be increased (see Figure 3-10). Therefore, many 0.25-micron processes support multiple voltage levels by defining a different minimum gate length for each different level. Changing the gate length can be easily achieved with ALM technology, but not so easily with the linear shrink approach. The reason is that, aside from the gate length, the gate width must also be adjusted and increased, which means that the entire layout design has to be modified.

![Figure 3-10](image)

*Figure 3-10. To operate a 0.25-micron design at voltage levels higher than 2.5 V, gate length must be increased.*

If the voltage level is reduced, dynamic circuit techniques must often be modified or substituted, which can mean that transistors have to be added, as shown in Figure 3-11. These modifications can be supported by an ALM technology that can support ECOs. In this case the circuit modifications can be done inside the layout without considering the final design rules. For example additional wires can be drawn with a smaller pitch or transistors can be added with smaller gate length and width. ALM can be used to adjust the layout to the correct design rules.

![Figure 3-11](image)

*Figure 3-11. Transistors must often be added if voltage level is reduced.*
Chapter 3

Lower voltage levels increase the vulnerability to signal integrity effects such as crosstalk. Wider power supply lines and wider spacing of critical signal lines can prevent these effects and can increase circuit speed.

3.3.2 Adjusting circuit speed

IP blocks – such as CPU cores, digital signal processor (DSP) cores, datapath, and memory – are used in very different applications. These applications can vary greatly in system and overall circuit complexity and require different circuit speeds.

To adjust a circuit to a different speed, designers must either do a re-layout if automatic place & route is used, or they have to modify the layout by other means, probably via manual redesign. The specific changes that are required for adjusting circuit speed are transistor sizing, adjusting wire width and spacing, and adjusting the power supply lines. Because these tasks must be done specifically for each device and wire, a general sizing of the whole circuit does not work.

In cases where the circuit speed should be changed by an order of magnitude or more, a total redesign of the circuit architecture may be necessary. In these cases, ALM technology cannot help.

ALM technology can be applied to the problem of sizing transistors from a netlist or similar input. It can be used to size the width of specific wires and space wires according to simulation results. This can greatly reduce design time in full-custom layout designs.

3.3.3 Low-power applications

Reducing the power consumption of circuits is desirable for several reasons. The biggest drivers for power reduction are mobile applications such as cellular phones and laptop computers. Others include cooling problems in high-speed devices, as well the environmental concerns of reducing overall energy consumption.

Some general methods of reducing power consumption are reducing voltage levels, reducing capacitance inside the circuit, reducing circuit speed, and the use of variable-speed devices and sleep modes in which activity is reduced to a minimum. However, sleep and standby modes do not reduce power consumption in the active mode. In general, circuit speed reduction is not desirable. Finally, reduction of voltage levels has been discussed above (see section 4.3.1).

Most often, the 80/20 rule holds for power consumption in IC design: 80% of the power will be consumed by 20% of the devices and wires. An effective way of reducing the power is therefore to concentrate on the 20%
of devices. The power consumption of a device is proportional to its switching frequency and to the gate and wire capacitance it is driving. Normally, switching speed cannot be changed, but capacitance can. A device’s fanout capacitance can be reduced by spacing wires more widely (see Figure 3-12), which will also allow gate width reduction. The reduced gate width will, in turn, reduce gate area, which contributes to gate capacitance.

Figure 3-12. Changing spacing and width of wires, as well as increasing W/L, can reduce delay time.

This process is already automated by different power optimization tools that are available in the market. The output of these tools are modified circuit netlists with modified transistor sizes. ALM technology can be applied to implement these new netlists by automatically adjusting the layout.

Figure 3-13. Additional mask levels and structures can often be added automatically with ALM technology.
3.3.4 Adjustments for different process options

Many circuits are developed for different process options, such as non-volatile processes for implementing EPROM, EEPROM or FLASH memory; processes that allow combining dynamic RAM and logic on one IC; BiCMOS processes; and processes that allow the implementation of high- and low-voltage transistors on the same silicon. These different options are often derived from a single process, but require additional mask levels and structures. These masks and structures can usually be added automatically using ALM technology (see Figure 3-13).

3.3.5 Customizing cell libraries

With ALM technology it is possible to modify and adjust all or part of a standard cell library for a specific purpose. Not only can they be adapted to the requirements of different technologies and speeds, libraries can also be optimized for a specific design or application. This can be done simply by optimizing only the cells that are really needed for that particular design.

3.4 Achieving Timing Closure

Timing closure has taken center stage as the toughest challenge of today’s deep sub-micron (DSM) designs. Consequently, timing analysis and timing verification have become most important. Timing verification and analysis is one part of the game, but what about fixing and correcting the detected problems? Many resources have been spent on integrating front-end synthesis and back-end physical design into an automatic, one-step process. Most approaches consider estimated physical information inside the synthesis flow. The problem is making exact estimations of the physical parameters without having a physical database available. This is a tough chicken-and-egg problem.

At the process technology level, the problem can be partly reduced by introducing other routing materials, such as copper, and providing better isolation between wires with low-k materials. This reduces resistance and capacitance of the interconnect wires, and reduces the delay ratio between transistor switching delay and wiring delay. However, this is only a temporary solution for a maximum of two process generations.

A more feasible solution is to correct the physical layout according to the findings of the timing simulation and analysis. Many people think this is impractical because of the lack of tools, but ALM technology can be easily used for these kinds of tasks.
### 3.4.1 The timing challenge in DSM technologies

Studying the timing problem in detail, it becomes obvious that the problem is not all that new. The cause of the timing closure problem is simple: the ratio of wire delay to transistor delay is increasing. Figure 3-14 shows the typical relationship and how it changes relative to the process feature size. This is not a problem per se. But under these conditions it is much more difficult to design high-speed circuits, and the classic synthesis-based design flow runs into limitations. This is because the synthesis approach only works well when the wire delays are very small compared to the gate delays, and/or when wire delays can be exactly estimated during synthesis.

![Figure 3-14. The timing closure problem is caused by the increase in the ratio of wire delay to gate delay.](image)

The fact that wire delays can be larger than gate delays is also not new. A very similar situation occurred when designs were done in single poly and single metal technologies. But at that time, all designs were hand-crafted layouts and nobody used synthesis and automatic place & route. Unfortunately, it is not an option to abandon the automatic synthesis and place & route tools of today’s design flows and substitute them with a full-custom approach.

The main reasons for the relative increase in wire delay are increased circuit complexity and faster transistors, changes in wire profiles in deep sub-micron processes, dynamic capacitance effects among wires, and reduced supply voltage levels.

Reduced process feature sizes have made it possible to increase the number of gates per design from several hundred thousand to several million gates. This means that the overall physical design size has remained the same, even when the typical feature sizes were reduced in area by a factor of
20 or more. Because of the absolute size of the circuit, the maximum wire length did not reduce to the same degree (see Figure 3-15). The consequence is that wire capacitance and wire delay time will increase in relation to the reduced gate capacitance and gate switching time of transistors.

**Figure 3-15.** Maximum wire length depends on die size and not on the process technology.

The second reason for the increase in delay time is the change in wire profile. To implement a narrow wire pitch without reducing the conductance of wires in DSM, the wire profile has changed from flat and horizontal to tall and vertical (see Figure 3-16). This has caused a significant increase in lateral capacitance between wires, which causes an increase in delay time.

**Figure 3-16.** To reduce the wire pitch in DSM design, without reducing conductance, the wire profile has changed.

DSM process technologies use smaller gate lengths and smaller gate oxide thicknesses to increase switching speed. These smaller gates must be operated with a lower supply voltage level, because a higher supply voltage level will damage the gate oxide and break the transistor. Therefore, the
supply voltage was reduced from 5 V in 0.5-micron technologies to 1.8 V in 0.18-micron technology. The signal delay time is also inversely proportional to the voltage difference: the higher the difference, the faster the change of the load and the shorter the delay time.

Increased signal frequency and greater lateral capacitance between wires are causing dynamic delay effects. The effective capacitance of a plate capacitor depends on the area of the plates, the distance between the plates, and their voltage differential. Usually one of the plates is on GND or VDD potential. In the case of neighboring signal wires, the area and the distance of the plates are given but not the signal level. Depending on whether the wires are working in the same direction or against each other, the delay time can be larger or smaller. This means the delay time is a function of the potential or signal level of the signals inside the circuit. The difference between the best and worst cases can be a high percentage of the total delay time in UDSM circuits (see section 8.1.1.1, Figure 8-2). This means that a circuit may fail for that reason. Unfortunately, the effort of simulating these effects for an entire circuit is very high and most often impractical.

The physical reasons for the relative increase in wire delay over transistor delay have been discussed above. The remaining question is why this is such a huge problem. A closer look at the design flow will help answer this question (see Figure 3-17). Most designs are done using a design flow containing synthesis and automatic place & route as the main elements of design. The design is described in VHDL or given by a schematic. The circuit description is refined to an RTL and gate-level description. Each level of abstraction is verified using simulation. During these simulations, no exact information is available about the placement of the elements and the routing between them. Since the wire delay has increased significantly in relation to the gate delay, simulation must consider these wire delays, usually via statistical approximations.

After simulation the circuit is physically implemented using place & route. The result is a layout database that can be used for accurate extraction of the real wire resistance and capacitance. These resistance-capacitance (RC) elements will then be fed back into the simulation for final verification of the design.

This flow worked well down to 0.25-micron technology and for a moderate circuit speed of up to 200 MHz. But for higher-frequency designs and smaller circuit dimensions, the number of critical signals will increase so much that this process may no longer converge.
3.4.2 Detecting timing problems inside a circuit

The most common method of detecting timing problems inside a circuit is by doing a timing simulation. To get the most accurate results, the parasitic RC elements from the wires are extracted from the physical layout. The RC elements are added to the netlist derived from the schematic or VHDL model. This new netlist will be simulated and the results of the simulation analyzed. The result of the timing simulation is a list of signals that do not match the timing requirements. These signals are called critical signals.

There are several key parts of this process. The first is extracting the resistance and capacitance of the wires from the layout. Depending on the required amount of accuracy, one-, two-, and three-dimensional approaches can be used. The one-dimensional approach uses wire length and area to approximate capacitance. The two-dimensional method also considers the lateral capacitance between neighboring wires by calculating the perimeter, and the three-dimensional approach adds consideration of the capacitance.
between the wires running on top and below (see Figure 3-18). However, none of these approaches considers the dynamic capacitance effects between wires. The computing complexity of a three-dimensional analysis is so high that it becomes impractical to consider it for full-chip analysis.

**Figure 3-18.** Extracting wire RC from layout can be done by 1D, 2D or 3D analysis, depending on the required accuracy level.

The next critical part is the number of RC elements by which the wire will be represented in the model. The minimum number of elements is one, but it is physically more accurate to represent a wire by a chain of RC elements, as shown in Figure 3-19. The more elements, the more accurate is the result. However, the CPU time required to simulate the circuit will increase significantly with the number of RC elements. Therefore, the RC elements are usually combined into one element where possible.

**Figure 3-19.** It is physically more accurate to represent a wire by a chain of RC elements.
For high-frequency circuits the extraction of RC elements is not sufficient: the inductance of the wire must be considered for dynamic effects. This means that resistance-capacitance-inductance (RCL) elements have to be extracted, which will increase the extraction and simulation effort.

For large circuits with hundreds of thousands of gates, simulation is not a practical solution. Instead, static timing analysis is used to do timing verification. The goal of static timing analysis is to estimate the worst-case timing path by adding up the delays of the elements in the network. For this purpose, the delays of every transistor, gate, and wire between the elements are calculated from the extracted RC values. Then the maximum delay times of all possible paths between inputs and outputs are calculated.

The result of static timing analysis is normally the upper boundary of the delay. This can be inaccurate for high-speed circuits, where inductance between wires will affect the result because inductance cannot be considered with static timing analysis. Inductance can affect the delay in both ways: it can either shorten the delay or increase the delay depending on the state of the involved wires.

Figure 3-20. Sample flow for extraction and analysis of critical signals.
To reduce the effort required for extraction and simulation, it is desirable to classify signals as critical and non-critical before doing a detailed analysis. For this purpose, the different extraction and analysis technologies can be combined by applying different methods to different classes of signals. For example, for uncritical signals a one-dimensional approach may be sufficient, but for critical signals the most accurate approach should be used. These techniques, implemented well, can significantly improve the performance of timing analysis. A sample flow is shown in Figure 3-20.

![Diagram](attachment:image.png)

*Figure 3-21. Relative comparison between source and target designs can substitute for a full timing analysis to ensure design performance after conversion.*

For ensuring the performance of a design after design migration, the relative comparison between source and target designs can substitute for a full timing analysis. Relative comparison means that the RC delay of each wire is extracted before conversion and then compared with the value after
conversion multiplied by a speed-up factor. The speed-up factor considers
the electrical relationship between the source and target technologies of the
migration. The final timing analysis will be done only for the purpose of
final verification. The flowchart for this process is given in Figure 3-21.

3.4.3 Changing transistor sizes

The result of a timing analysis is a report of timing errors. Timing errors
are either setup time or hold time violations. Setup time violations can be
corrected by reducing the signal delay. Hold time violations can be corrected
by increasing the signal delay. In the past the majority of problems were
setup time related. In extreme fast technologies we find more and more hold
time violations.

The result is a list of signals that do not make their delay targets. The
next step is to correct these errors. Most approaches consider re-routing the
circuit, re-layout, transistor sizing, or driver insertion. The approach of re-
routing or re-layout is only useful when the timing is totally out of order.
However, there is no guarantee that the new layout will have better timing.
At a certain point of complexity, every result will have timing errors. The
first step in fixing these timing errors is to change the transistor sizes on the
critical delay path. Most tools only enlarge the gate width of the relevant
driver transistors. Other approaches also reduce the sizes of transistors that
are part of the load of the critical delay path. This reduces the load of the
driving transistors, and speeds up the signal. The combination of enlarging
and reducing transistor sizes also improves power consumption.

![Figure 3-22. Different ways of sizing transistors.](image)
Often the layout topology does not allow simply increasing the width of the transistor. In many cases, fingers must be added or deleted to implement the correct gate size. This is a complex operation that can only be achieved by applying sophisticated layout modification technology. Figure 3-22 illustrates the different possibilities.

In other cases, it may not be sufficient to increase a single driver; instead, it may be necessary to insert an additional driver. This operation is even more complicated and may require a local re-layout of the circuit.

3.4.4 Modifying wire capacitance and resistance

Since the main source of capacitance in DSM designs is wire capacitance, it is possible to modify the load of a gate by modifying the wire capacitance at the output of the gate. One way to achieve this is by a re-layout with shorter wires. This may be difficult and create more new delay problems than it solves. However, a more elegant method is to modify the spacing of critical signals. Doubling the wire spacing reduces the fringe capacitance of the relevant wire portion by a factor of about two. If this could be automatically achieved by ALM technology it would be of great benefit to circuit design, because it would reduce the delay of the critical wires without sacrificing too much area. It would also reduce power consumption, heating problems, and metal migration problems. Figure 3-23 shows how this looks in principle on the layout level.

![Diagram](image)

**Figure 3-23.** Doubling wire spacing reduces lateral capacitance by a factor of two.
Similar results can be gained by modifying the resistance of the wires. Reducing wire resistance can be achieved by increasing the width of wires or by adding a bundle of wires for specific signals. Reducing this resistance will not significantly affect capacitance, and thus will not affect power consumption, but it will prevent metal migration. Therefore, it is most useful for power signals, clock signals, and very fast high-load signals.

3.4.5 Clock optimization

A limiting factor for circuit speed is clock skew, which is the delay difference between different end nodes of the clock tree. The basic idea of a clock tree is to match all delay paths from the clock source to each gate as much as possible. The higher the clock frequency, the more critical and difficult it is to achieve a small clock skew.

In synchronous DSM designs, the clock tree needs to be even more carefully designed. It is no longer enough to create equal-length and wider-than-minimum branches for the clock tree, because the delay is highly influenced by the fringe capacitance between the clock and other signals. This means that clock signals have to be shielded from, or spaced more widely to, other signals.

A better and more accurate solution would be to take the detailed capacitance and resistance of every branch of the clock tree into account. By making a detailed analysis and specifically changing the width and spacing of each branch of the clock tree, a better clock skew optimization is possible.

3.5 Solving Signal Integrity Problems In DSM Circuits

In DSM circuits, signal integrity effects can lead to circuit malfunction. ALM technology can be used to reduce and eliminate signal integrity problems on the layout level, specifically by reducing noise and crosstalk.

3.5.1 Reducing noise

As coupling capacitance and inductance increases in DSM circuits, crosstalk becomes a serious problem. Crosstalk is the propagation of a signal to a neighboring wire. If not properly handled, it can degrade signal integrity and cause circuit malfunction when the coupled noise exceeds allowed thresholds.

Another potential source of noise derives from simultaneous switching. When many gates switch at the same time, the resulting current draw causes a voltage fluctuation on the power and ground network. If this fluctuation
makes the voltage levels on the network exceed allowed thresholds, malfunctions can occur.

![Flowchart diagram](image)

*Figure 3-24. Flowchart of an approach to filtering signals critical to crosstalk and voltage drop effects.*

Crosstalk and voltage fluctuations due to simultaneous switching can be detected by detailed electrical simulation of the circuit, considering the RCL elements of the relevant wires. To do such a simulation for all wires in a multimillion gate device is computationally impossible. Therefore, intelligent filters have to be used to detect signals which may be exposed to these effects. For designing such filters, the following observations can be made:

- **Crosstalk increases with signal frequency.**
  The maximum signal frequency or the frequency range of a signal can be analyzed during logic simulation. Only signals above a certain frequency must be considered as critical.

- **The danger increases with the length of parallel wires.**
  For every frequency there is a specific critical length for parallel wires that also depends on the voltage level.
Crosstalk increases with the reduction of supply voltage levels.
High-voltage circuits are much more immune to crosstalk effects than are low-voltage circuits.

Voltage drop increases with the total capacitance connected to simultaneously switching gates.
The amount of voltage drop can be estimated from average gate switching frequencies and from gate and wire capacitance. Switching frequency can be derived from logic simulation.

From these observations, an effective filter algorithm for signals critical to crosstalk and voltage drop effects can be derived. These critical signals can be further analyzed to determine whether a real problem exists. Figure 3-24 shows a flowchart of a principal filtering approach.

3.5.2 Reducing crosstalk

Crosstalk cannot be reduced by performing transistor sizing or netlist modification. Instead, to reduce or eliminate crosstalk effects, the wiring of the relevant signals must be re-laid out or modified. The most practical solutions are:

- Spacing critical wires farther apart to reduce cross-coupling capacitance between the signals. This is the easiest and most efficient method. The spacing of specific wires can be automated and integrated into an ASIC design flow.
- Shielding the wires with GND lines or VDD lines. This is the second-most effective way. This method will reduce the dynamic capacity effects that are caused by simultaneous switching of neighboring wires. These dynamic effects can increase capacitance by a factor of two or more. With shielding, the signal is surrounded by a static signal, either VDD or GND. This will not eliminate the effect, but it will reduce the effects of capacitance in a dynamic state to a minimum.
- Reordering wires to reduce dynamic cross-capacitance effects. This is the most sophisticated approach. It is used most often on bus structures where the signal and the inverted signal are running in parallel to each other. This method will also create dynamic capacitance, because the signals are always switching simultaneously. Therefore, it is better to exchange the order of the wires in the bus as shown in Figure 3-25.

Most applications use one of these three methods. In very critical applications, it may be necessary to combine all three for a particular pair or set of wires.
3.6 Design Optimization

Design optimization is usually an afterthought in the design process. It is almost never done because of the lack of resources and the pressure to go to production and tapeout. However, in some cases design optimization is an economic requirement for making a design profitable in the market; or it can produce higher production yields; or it can create superior differentiation from a competitor's product, such as higher speed, lower cost, lower power consumption, higher reliability, or longer product life cycle. All design optimization requires layout manipulation, which is too time-consuming and error-prone when done manually. ALM technology provides a way of automating most of the necessary tasks.

3.6.1 Timing optimization

The primary benefit of timing optimization is higher circuit speed. Besides the obvious advantage that speedier circuits often sell better and for higher prices, higher speed can also increase the performance of the circuit over a larger range of operating temperatures. This will make the circuit suitable for a greater number of applications and will therefore increase its marketability.

The technical process of timing optimization consists of detecting timing bottlenecks, finding a cure, and solving the problem. The bottlenecks can be found by timing simulation, simply by increasing the target speed until the circuit fails. The cure is usually a combination of device and wire sizing. The specific wire and transistor sizing can be implemented using ALM technology.

3.6.2 Power optimization

Power consumption is a critical decision factor in most applications. Reducing the power consumption of a circuit beyond the formal specification may increase the circuit’s market and raise its price. Power optimization can be divided into two sub-problems: reducing power
consumption and improving power distribution. Reducing power consumption is critical for the mobile and automotive markets. In addition, it has the positive side effects of reducing heat dissipation and of reducing long-term reliability problems such as electro-migration effects on power lines.

Improving the circuit’s power distribution will reduce the danger of hot spots, as well as reduce the need for external cooling devices. This requires simulating the power consumption or maximum current on the different wires of the power grid. After detection, the solution is to improve the power grid, which means adding more supply lines or widening existing supply lines. ALM technology can be used to support both layout tasks.

Reducing overall power consumption can be achieved by power optimization circuit analyzers. These tools use the extracted netlist from the layout and make a detailed simulation to find out where transistor sizes can be reduced to reduce power consumption. Since power consumption can be calculated as the sum of all capacitances multiplied by the switching speed, reducing gate size will reduce gate capacity and therefore reduce power consumption. Gate size reduction is only done if it will not constrain the specified target circuit speed. An additional way of reducing power consumption is by reducing capacitance between wires with high switching frequencies. This will affect static and dynamic capacitances, as well. Instead of reducing maximum circuit speed, this method of reducing power consumption may increase it.

Changing transistor sizing and wire spacing according to the analysis of power optimization tools can only be handled by automatic tools, because of the huge number of transistors involved. Who wants to change 100,000 transistors in a one-million gate design, for example? ALM technology is an approach which can implement such changes in the layout and can help to implement automatic power optimization.

### 3.6.3 Yield optimization

Yield optimization becomes an increasing issue in UDSM technologies below the 0.18-micron technology node. Some industry sources claim that the gap between the obtainable yield of a yield-improved design and the yield of an unimproved design will exceed 20% at 0.13 micron and below.

The reason for this trend is that the number of critical features in a design has increased at a faster rate than has the reliability factor of producing a specific feature. Reliability is defined as the number of errors divided by the number of features produced. For example, assume that processing contacts is so reliable that only 1 out of 1 million contacts is faulty. This means that, in a design containing 100,000 contacts, an average of 1 out of 10 chips on a
wafer may have a fatal contact error. If the chip with 100,000 contacts has about 50,000 gates, then a chip with 10 million transistors would have about 20 million contacts. To get the same 90% yield, the reliability factor for processing a contact must increase from 1 error out of 1 million contacts to 1 error out of 200 million contacts. This means the reliability of processing a contact must increase by a factor of 200 to produce the same yield for this feature.

It is clear that with a continually growing number of features, reliability must be increased either by improving processing reliability or by other means, such as increasing redundancy or relaxing critical features.

Redundancy will automatically increase reliability. The best examples of redundancy are duplicated contacts and via holes. For example, when every contact is duplicated, the chance of a contact failure will be reduced from $1/n$ to $1/n^2$ if the error probability for single contacts is $1/n$. Therefore, some fabs have introduced procedures to create duplicated via holes for their latest process technologies below 0.18 micron.

The other way to decrease the probability of errors is to make critical features less critical. Examples are increasing transistor endcaps, increasing contact-to-gate spacing, increasing overhangs over via holes and contacts, and increasing wire spacing and width. All of these enhancements would require more layout space if they were applied to all structures. However, increasing die size would reduce the number of circuits per wafer and also reduce yield. A good compromise is to selectively implement improvements in the layout without increasing overall layout size.

As the number of critical features in a design has grown to tens of millions or more, an automatic approach like ALM technology is needed. By applying a modified compaction/decompaction approach, ALM technology can perform all of the above-mentioned tasks most effectively, without inserting additional design rules into the layout or flattening the database.

### 3.6.4 Electro-migration

Electro-migration is a long-term reliability problem: it can produce circuit failures after years of use. Such circuit failures are critical in sensitive applications like airplanes, brake systems in automobiles, or health applications, where long-term reliability is a must.

The critical wires where electro-migration can occur are high-speed, high-current signals, like clocks and high-speed busses. Normally, the solution to this problem is widening the wire. This can be done with the help of ALM technology, once the critical signal has been identified.
3.7 Optical Mask Enhancement

Advanced process technologies below 0.18 micron require advanced optical processing, such as optical proximity correction (OPC) or phase shift mask (PSM) creation. OPC and PSM require layout modifications, which increase the number of vertexes in the layout. Because they often have to be executed on the flat layout database, they increase file size and are a major factor for the increase in mask-making costs.

3.7.1 Optical proximity correction

In every optical or printing process the structures on the wafer differ from the structures on the mask. An example of these differences is given in Figure 3-26. Here, sharp corners on the mask become round on the wafer, or small gaps are filled. With the decrease in the minimum structure size and the unavailability of light sources with much smaller wavelength, these effects can cause fatal circuit errors. Such errors on the wafer include shorts and opens, and structures that do not overlap as they did on the mask. Even when the effects are not fatal to all die on the wafer, they will significantly reduce the wafer’s yield and make economical production of the circuit impossible.

Optical proximity correction modifies the mask so that the image on the wafer is similar to the original image on that mask. The mask is modified so that the proximity effect is compensated for on the wafer (see Figure 3-27). So far, this solution is workable; however, the number of vertexes in the resulting new mask can increase by a factor of 2 to 5. This increase creates additional costs for the development and production of new designs, and will result in higher mask costs that may, in the near future, reach $2 million (US) per design.

There are two commercial OPC methods: image simulation and rule-based. Image simulation is more expensive but creates fewer vertexes. This method simulates or calculates the image on the wafer and then checks...
whether OPC is needed for that structure. This process is slower, since it is more CPU-intensive, but it reduces mask costs by creating less data.

The second approach is rule-based. Based on the definition of critical structures or patterns, material is either added to, or eliminated from, the structure. Since no check is done in the rule-based approach, even if it is needed in a specific situation, this approach creates more data than does the image simulation approach.

![Proximity Corrected Layout and Printed Wafer](image)

*Figure 3-27. Optical proximity correction modifies the mask to produce an image on the wafer more like the original mask image.*

![Diagram of Layout and Structures](image)

*Figure 3-28. Making structures uncritical where possible reduces the amount of OPC required.*
3.7.2 OPC avoidance

A third, more economical approach would be OPC avoidance. OPC avoidance makes structures OPC-uncritical where possible, thus reducing the amount of OPC required. Figure 3-28 shows the example of the end-of-line rule between contact and metal. The top part shows the end-of-line rule without OPC and the structure printed on the wafer. The wire overhang over the contact is shorter and the connection may be lost if some misalignment occurs. If the metal is extended over the contact a little bit more, as shown in the middle part of Figure 3-28, the connection between wire and contact is ensured, even when the proximity effect shortens the wire. This approach is more sophisticated and complicated than the first two approaches.

Figure 3-29 shows an example with two cells. Cell A is placed to the left of cell B, and cell B is placed again to the right of the first placement of cell B. After the correction is done, the result looks good in the far right placement of cell B, but in cell A and in the middle placement of cell B, it creates a design-rule violation and it might create a short. The solution of the problem is to do the correction only for cell B and correct the situation in cell A by applying OPC.

![Figure 3-29](image)

*Figure 3-29. In this example, only the far right placement of cell B is corrected, and OPC is used to correct the resulting problem in cell A and in the far left placement of cell B.*

Figure 3-30 shows how solutions for a small structure may differ using different approaches. Figure 3-30a has no OPC correction. Figure 3-30b shows a rule-based solution creating 84 vertexes. Figure 3-30c shows OPC avoidance by applying design rule relaxation which reduces the vertexes to 52.
3.7.3 Phase shift mask creation

The main technology used for resolution enhancement is the creation of phase shift masks, which can be automatically produced by tools available on the market. A phase shift mask for gates is created by adding two polygons on each side of the gate. The two polygons represent the 0- and 1-phase masks. The masks have to be created so that a 0-phase mask will not touch another 0-phase mask or a 1-phase mask another 1-phase mask. This is relatively easy to achieve when the only PSM masks created are those for gates, as shown in Figure 3-31. This Figure shows the PSM of a gate and how the image looks on the wafer. This is a technique for producing transistor gates with smaller gate lengths, even when the poly width has to be a larger size. Creating a PSM mask for an entire poly or metal layer is very difficult because of phase shift conflicts.
In Figure 3-32 a T-shaped polygon is drawn. On one side of the T is a 0-phase PSM polygon, and on the other side is a 1-phase polygon. The question now is which mask to put at the top of the T. Neither a 0- nor a 1-phase mask here will satisfy the condition that a 0- or a 1-phase mask may not touch another 0- or 1-phase mask. This situation is called a phase shift mask conflict, or odd cycle. The solution for this problem is to make one part of the T non-critical regarding PSM. In this case, the width of one part of the T must be increased. In large layouts hundreds of thousands, or even millions, of such conflicts can exist. This is why most PSMs are created only for gates and not for the entire poly layer or metal mask. For further technology reduction, PSM must be applied to the entire layer. To do this, the conflicts must be detected and solved by manipulating the mask layout in a non-trivial way. For this task, ALM technology or compaction technology can be applied.

![Phase Shift Masks](image)

*Figure 3-32. Phase shift conflict with possible solution.*
Chapter 4

Characteristics and Functionalities of an Automatic Layout Modification Tool Suite

This chapter discusses the characteristics and functions which should be provided by an automatic layout modification (ALM) approach, and which are needed for mask layout correction, migration, and optimization. Because technology, design requirements, and applications continue to change, the properties discussed here are not complete, nor do they comprise the minimum requirements of such a tool. Many of these properties and functions are already implemented separately in other tools and none of them are conceptually new. However, combining these functions in a single software tool for ALM is new, and provides faster processing speed and a shorter design flow without the need for interfacing between separate tools. Some applications are only made possible by integrating some of the functions in a single tool, because internal data cannot be accessed if the functions are implemented in separate tools.

4.1 Migrating the Electrical Characteristics of a Layout

One of the major differences between a linear shrink approach and automatic layout modification is that ALM considers the electrical characteristics of the target technology. In order to do so, a sophisticated approach for sizing devices must be available. Devices that require sizing include different types of transistors, such as N-, P-, pull-up, pull-down, high-voltage and low-voltage transistors, as well as resistors, capacitors, and
diodes. The software should allow designers to specify user-defined special structures, such as probes and pads, which are not in themselves electrical devices, but which require special sizing and different design rules. Some electrical devices are not even part of the schematic or the netlist, for example, wires, contacts, and vias. Physically, wires and contacts are passive devices and have to be modeled as capacitors and resistors. But in earlier process technologies down to 0.35 micron, they could be ignored because they were electrically irrelevant, due to their low electrical values compared to active devices. However, in the latest process technologies these “passive” devices are becoming increasingly important, and, if not considered during design, can cause circuit failures. Since electrical parameters translate into geometrical structures, and modifying the size of these structures changes their electrical values, an ALM approach should be able to distinguish wires and via cuts as devices used explicitly to control the capacitance and resistance of signals. Figure 4-1 shows how the number of via and contact cuts are increased to reduce resistance during a design migration.

![Figure 4-1. Reducing resistance by increasing the number of contacts and via cuts.](image)

Designers should be able to specify target device sizes in a variety of ways in order to handle all possible cases and applications. The most common way is to link the devices to a netlist or schematic, and import the device sizes from there. This can be done with all devices defined in the schematic and the layout design, so they can be identified by comparing the layout with the schematic or netlist. For devices that are not part of the electrical netlist, such an approach is impossible. These devices include all parasitic devices, such as wire capacitance and contact resistance. They can be sized using a sizing function that describes the relationship between source and target electrical values. These sizing functions can also be used if no updated schematic or netlist is available, or if the design flow does not
have the capability of doing layout-versus-netlist comparisons. Figure 4-2 shows examples of sizing functions for transistors, wires, contacts and capacitors.

![Diagram](image)

**Figure 4-2.** Examples of sizing functions for transistors, wires, contacts and capacitors.

Another direct way of controlling device sizes is by placing a text string, which states the new size, into the mask layout. This text will be picked up by the tool and interpreted as a sizing instruction. This approach works especially well for, and is needed by, wires that have different widths over their total length. Figure 4-3 shows an example with text instructions.
Chapter 4

Since wire capacitance is also a function of wire spacing, the sizing function of an automatic layout modification approach should also make it possible to define signal-specific wire spacing, as shown in Figure 4-4. This is very important when considering signal integrity problems such as crosstalk.

Figure 4-3. Example of text instructions for controlling device sizes.

Figure 4-4. Sizing function of an ALM approach can define wire spacing to control capacitance.
4.2 Signal-Specific Layout Manipulation Support

The combination of higher signal speed, lower voltage levels, and higher wire capacitance causes signal integrity problems, the major effects of which are increased signal delays or incorrect signal levels. Methods for detecting and locating these problems are discussed in Chapters 3 and 8. After detection and location, the problem usually must be solved on the physical level.

To be able to solve signal integrity problems on the physical layout level, an ALM tool must provide signal-specific layout manipulation functionality. The same functions can also be used to solve timing problems by modifying wire spacing and width.

4.2.1 Solving signal integrity problems

The main ways of solving signal integrity problems on the physical level are:

- Separating signals by re-routing all or part of the circuit.
- Increasing the spacing between critical wires.
- Shielding wires with VDD or GND wires.
- Changing the order of bus wires.
- Moving wires to another routing level.

![Re-arranged Signal Lines](image)

![VDD/GND ShieldingWires](image)

*Figure 4-5. Examples of making local corrections to signal integrity problems by changing wire spacing, bus wire order, or metal level of wires, or by inserting GND or VDD wires for shielding.*
Because signal integrity problems are usually detected in the later phases of the verification cycle after place & route and timing verification, a local correction approach is desirable to prevent the need for a total re-route and re-verification. This can be achieved by changing wire spacing, locally reordering bus wires, moving wires to a different metal level, or inserting GND or VDD wires to shield them and provide a defined voltage base. Examples of such modifications are shown in Figure 4-5.

Signal-specific wire sizing and spacing of critical signals, along with shielding and moving signals to separate metal levels, are especially helpful functionalities for supporting signal-specific layout tasks with an automatic layout modification approach.

4.2.1.1 Signal-specific wire sizing and spacing of critical signals

Designers should be able to define the wire width and spacing of critical classes of signals. Critical signals can be signals, which are too fast or too slow or are effected by signal integrity effects. These signals should be separated from each other by specifically defined design values that are different from the design rule values in the design rule manual, which only specify minimum requirements. It would be especially desirable if these values could be specified differently for the different levels of metal, because of their different layout density and wire length, in order to optimize for the smallest layout. To do so, an ALM tool should provide a method for tracking signals on a global basis inside the layout. A form of netlist extraction from the layout must therefore be available.

Examples of these critical classes of signals are:

- Supply signals, such as VDD and GND.
- Clock signals.
- Bus signals of different polarity, such as b and not b.
- High-speed signals.
- Analog signals.

A class of signals can contain one or more specific signals. The following statements are examples of specific design rules using classes:

- The spacing between the class SUPPLY and the class CLOCK should always be greater than 3 times the minimum spacing.
- The spacing between high-speed signals S and CLOCK should be greater than 0.5 micron.

The statements above can be shortened to:
SUPPLY := VDD, GND;
CLOCK := CLK, CLK;
SPACING SUPPLY – CLOCK >= 3 x metal_spacing;
SPACING S - CLOCK >= 0.5

Often, wires on the upper metal layers run long and in parallel, but on the lower levels like metal 1 and local interconnect, they run shorter. Therefore, it is also useful to be able to define signal-specific spacing for specific layers, such as for metal 3 alone, or for metal 2 alone.

**4.2.1.2 Shielding and moving signals to separate metal levels**

Another way of separating signals that affect each other is to move one of them to a different metal level, as shown in Figure 4-6. This requires extensive signal-specific mask operations to ensure circuit correctness. For higher automation, this procedure has to determine whether there is space available on the other mask level, and insert the necessary vias to connect the wires.

As shown in Figure 4-5, shielding is a similar task, in which additional wires are inserted between existing wires.

![Figure 4-6. Separating signals that affect each other by moving wires to a different metal level.](image)

These two methods of signal-specific wire treatment, shielding and moving wires to different metal levels, can also be combined for greater effectiveness.

Merely increasing drivers or reducing capacitance is usually not sufficient for very large nets. In many cases, such as a clock net, the resistance of these nets must be reduced by increasing wire width. Therefore, ALM software should be able to support the manipulation of non-minimum width nets, or even of nets whose parts have different widths. Extracting wire width is not a simple problem, as shown by Figure 4-7. Even if the metal 2 rectangle has the same dimensions in both cases, the electrical width of the polygon is longer than its physical width in one case, and shorter than its physical width in the other case. To find the correct width, the direction of current flow must also be identified.
The situation with power wires is usually the opposite of that just discussed for signal wires. Since power wires do not change their voltage level as signal wires do, there are no speed issues. Therefore, it is usually preferable that power nets contain a lot of capacitance. A high capacitance on the power net prevents power drops when many transistors are switched at once. For this reason, chip area that contains only routing and no active devices is often filled up by capacitors to reduce the danger of power drops. These capacitors are constructed as shown in Figure 4-8. They are built using poly and diffusion layers, with the poly connected to VDD and the diffusion connected to GND, or vice versa. Another way of reducing current-resistance (IR) drops is to reduce the resistance of power wires by increasing their width. To ensure and optimize power distribution, wide metal extraction and sizing capabilities are needed. Furthermore, selectively widening power wires or automatically inserting additional de-coupling capacitors on the power nets are very useful functionalities.

*Figure 4-7. Extracting wire width requires also identifying current flow direction.*
4.3 Ensuring Layout Quality Throughout an Automatic Layout Modification

When layouts are modified automatically, most engineers will be concerned about layout quality. However, layout quality means different things to different people. For some, quality is expressed by a very regular layout with an orderly and symmetrical look. Others do not care about appearance, but want to achieve the highest yield in the smallest area and with minimum power consumption. Since it is impossible to define a single push-button approach that fits everyone’s definition or priorities, a layout modification method should provide enough functionality to support most designers’ target specifications.

Some of the basic functionalities needed to accomplish this are a tool’s ability to control the number of notches and jogs; to handle non-orthogonal structures; to improve connectivity, latch-up characteristics, and the power grid; and to apply fabs’ recommended rules.

4.3.1 Controlling the number of notches and jogs

A very high number of notches and jogs in the layout structure is undesirable for several reasons:

- They do not look orderly or symmetrical.
- They increase the database size.
- They produce higher mask costs because of the increased number of vertexes.
- Verifying such layouts requires more computing resources.
- Modifying such layouts requires more effort.
However, the use of jogs is often necessary for creating a small and correct layout. Therefore, a layout manipulation program should be able to control the number and sizes of additional jogs, to insert them only where necessary, and to eliminate unnecessary jogs and notches from the layout database. In addition, the tool should be capable of the following:

- Guaranteeing a minimum length of new edges.
- Inserting notches only if they are larger than minimum size.
- Requiring a minimum length of edges where a jog may be inserted.
- Controlling orthogonal or non-orthogonal (45-degree) jogs.

Figure 4-9 shows the result of using layout modification with and without the use of jog minimization approaches.

![Layout modification with jog minimization approach](image)

![Layout modification without jog minimization approach](image)

*Figure 4-9. Jogs are often necessary to create small, correct layouts, so layout manipulation tools should be able to control them precisely.*

### 4.3.2 Handling orthogonal and non-orthogonal structures

Another criterion for ensuring layout quality with ALM technology is the ability to handle non-orthogonal, 45-degree structures. They can reduce the layout’s area, but often they are also needed for specific structures, such as meander-shaped transistor gates, where 90-degree corners are not allowed at all by the most recent process technologies. The use of 45-degree structures is very likely to increase along with the increase in higher-frequency
designs. This is because 45-degree wire corners are less likely to reflect electrical waves than are 90-degree corners. In high-voltage designs, with a supply voltage of 24 V to 500 V, 45-degree wire corners are commonly used to prevent high electrical fields at 90-degree wire corners. To efficiently modify 45-degree structures, the ALM tool should be capable of the following:

- Converting 90-degree structures to 45-degree structures and vice versa where desired (see Figure 4-10).
- Preventing 45-degree structures from being converted into orthogonal structures where not desired.
- Correctly measuring design rules for 45-degree structures (see Figure 4-11).
- Converting 90-degree gates into 45-degree gates (see Figure 4-12).
- Controlling the minimum and maximum length of 45-degree edges and structures.

![Figure 4-10. Converting 90-degree structures to 45-degree structures.](image)

![Figure 4-11. Different design rules for width and spacing for 45-degree structures and orthogonal structures in UDSM technologies, e.g. s* > s and w* > w.](image)

![Figure 4-12. Converting 90-degree gates into 45-degree gates.](image)
4.3.3 Improving connectivity

The design of connectivity can have a major impact on a design’s performance. As structures become smaller in UDSM designs, the electrical resistance of contacts and vias becomes a greater concern. In order to reduce resistance, a larger minimum number of contacts and vias for each connection is often needed. However, it is also desirable to increase the number of vias and contacts beyond the minimum requirement, wherever space is left in the layout. This increase will benefit the speed of the design as well as its reliability and yield. Some semiconductor vendors have therefore introduced automated mask manipulation procedures that systematically increase the number of redundant vias and contacts.

To control the number of contacts and vias, the following functionality should be available:

- Controlling the size and number of vias and contacts for each connection without violating design rules (see Figure 4-13).
- Maximizing or minimizing the number of via cuts and contacts without violating design rules (see Figure 4-14).
- Splitting large contacts and via cuts into smaller ones while maintaining design-rule correctness (see Figure 4-15).

![Figure 4-13. Controlling size and number of vias and contacts for each connection without violating design rules.](image)

![Figure 4-14. Maximizing or minimizing the number of via cuts and contacts without violating design rules.](image)
4.3.4 Improving latch-up characteristics

Circuits in UDSM technology become increasingly sensitive to latch-up failure. In order to reduce this sensitivity, designers can use epitaxial wafers and/or improve the substrate and well connections inside the design. Although epitaxial wafers cost more, they are now widely used. However, special care must be taken with substrate and well connections. When changing process technologies, the requirements for these contacts often become more restrictive—meaning that the maximum distance between substrate contacts and transistor gates is shrinking—and more contacts must be inserted. In addition, redundant contacts reduce latch-up sensitivity and increase yield.

Therefore, ALM technology often provides the ability to increase or reduce such structures to control their size. The optimum solution is to fill all of the remaining, unused space in the layout with substrate and well connections. This can be done in a manner similar to the way that contacts and via cuts can be maximized. Figure 4-16 shows a layout before and after improvement of the substrate diffusion area.
4.3.5 Improving the power grid

Because of higher circuit speeds and lower voltage levels, the power grid in UDSM designs becomes increasingly sensitive. To improve the power grid above the minimum requirements, the following layout manipulations can be made:

- Increasing the width of power busses where space is available (see Figure 4-17).
- Increasing via cuts between power wires to reduce resistance.
- Inserting capacitors on power busses to reduce the danger of IR drops during operation (see Figure 4-18).

These changes should be done without creating additional design-rule violations. The specific design rules to be considered are:

- Maximum width rules for wide metal layers.
- Wide wire spacing.
- Automatic insertion of wire slits in very wide metal wires.
- Wire width-dependent coverage of metal over vias.
- Double via requirement for wide lines.
Many of these rules are difficult to enforce with the most commonly-used design tools. Tasks such as wire slitting, for example, are still done manually in many design groups. Wire slitting requires knowledge of the direction of electrical current flow, since a blind slitting of wires can be fatal: when the wire is slit in the wrong direction or at the wrong place the resistance can be increased and the effective width is reduced. Figure 4-19 shows a wide metal wire with correct and incorrect slitting.
Many wafer fabs provide so-called “recommended rules.” The idea is to implement them where space is available without enlarging the total layout area. Since manual implementation is too time consuming and too expensive, an automatic approach is needed. ALM technology is the key to implementing those rules which should be applied to certain critical parts of the design to increase yield. From the viewpoint of process technology, the minimum design rules are often not ideal. Instead, they tend to be overly aggressive, either because of marketing requirements or because a linear shrink path for older designs must be provided and cannot be done with less aggressive rules. Examples of such rules are contact-to-gate spacing or poly overhang diffusion (endcap). Even a small relaxation of these rules can have an impact on yield. Figure 4-20 shows a small layout example before and after applying recommended rules or relaxed rules. Even in the places where no space is available for the full recommended rule, the layout is changed towards the recommended rule as much as the space allows.
To implement recommended rules inside a design, two problems must be solved. First, the size of the overall layout should not increase. Second, priorities for the different rules must be set. For example, two recommended rules are given, one for the spacing of metal and one for its width. But in the given area, only one rule can be implemented. Which one should be given priority: spacing or width? An increase in width will reduce resistance and may relax or solve a phase shift mask conflict. A larger space between wires will reduce wire capacitance, and may solve a timing problem and reduce power consumption. Thus, the problem is one of multi-level optimization. An exact solution for this problem can require exponential computing time, which is not practical. Figure 4-21 shows an example with different solutions for design rule relaxation depending on the set priorities. Part a) shows the layout without improvement. Part b) shows it with increased contact overhang and increased endcaps, and part c) shows it with enlarged power width to maximize power net capacitance.

To solve this dilemma, the rules must be weighted and prioritized. For example, phase shift conflicts should be solved before timing problems are solved, and timing problems before power consumption problems.
Some recommended rules do not affect the electrical performance of the circuit, but do improve yield, for example, the contact-to-gate spacing rule. However, contacts are connected to metal, so moving a contact will also impact the metal layer.

Implementing recommended rules inside a mask layout is a complex task. To support it, ALM technology should provide the following functionality:

- A language for describing recommended rules.
- The ability to prioritize and weight the rules.
- A rule-based system for applying specific rules.
- The ability to apply the rules only to defined areas of the layout.

4.4 Modifying Libraries Automatically

The modification and conversion of standard cell libraries is a major application for physical design reuse and ALM technology. The need for fast library generation is increasing because of early access to advanced technologies and more complex SOC designs, which require multiple libraries for high-speed, low-power, and high- and low-voltage circuits. In addition, many physical design tools such as place & route require more views of similar cells in order to guarantee timing closure. Some library-specific functionality is needed for efficient automatic library modification.
4.4.1 Processing an entire library

Standard cell libraries contain several hundred individual cells. Adjusting all of them manually in order to migrate them to a different process would be a huge effort. The major portion of this effort is adjusting the mask layout. To automate this work, an ALM tool should provide a way of automatically processing an entire library. This can be done by providing a scheduler that distributes the cells to different CPUs in the network for processing. The setup for the technology and the control parameters should be developed and tested on a small set of sample cells.

4.4.2 Supporting multiple grids

Most place & route tools work more efficiently when they use a grid-based approach instead of a gridless approach. The tools also work more efficiently if the number of grid points is small. Thus, designers will choose the largest possible grid without wasting silicon area for the place & route tool. The largest possible grid is usually the routing pitch, which can be different for different layers. Layout manipulation software must therefore support different grids for different layers and purposes.

![Figure 4-22. Principal structures of a standard cell, showing grids and pick-up positions.](image)

Figure 4-22 shows the principal structure of a standard cell with the different grids and pick-up positions. For example, it is often necessary to
support different grids in the x and y directions, and the grid must be centered for some structures, such as via cuts or port positions. To support all of these different requirements, an ALM tool should provide the following functionality:

- A different grid for each layer.
- Ability to apply the grid to the center of a pick-up position.
- A different grid in the x and y directions.

### 4.4.3 Controlling power bus width and cell height

The architecture of standard cells usually demands a common cell height and a common power bus width. Some libraries also work with single- and double-height cells. To achieve and enforce these needs, ALM technology should provide functionality to implement and support this requirements. Besides the width of the bus, there are different bus schemata that may need support, ranging from straight busses with a single width to banded ones with variable widths. Figure 4-23 shows several examples. Common to all structures is the fact that the ports are in the same y position, so abutting the cells is possible without violating design rules.

![Figure 4-23. Different power bus schemata may need support from ALM design rules.](image)

Two strategies can be applied to ensure a common cell height. One strategy is to run all of the cells and let the cell with the maximum height define the common height. The other way is to set a fixed cell height and
optimize those cells with heights that do not match it. The second approach is usually chosen for standard cell libraries. In this case, a good analysis tool can help to identify the cells that are too high. These cells are then modified or run with a different set of parameters.

### 4.4.4 Creating pick-ups

The latest generation of place & route tools does not require explicit ports inside standard cells. Instead, these tools dynamically extract the optimum position for ports. When standard cell libraries are modified, the task is to ensure a minimum of one possible port position, also called “pick-up”, for each signal. Figure 4-24 shows a cell with and without pick-ups.

![Figure 4-24. The same cell with and without pick-ups.](image)

To improve the library’s routability, additional redundant port positions can be inserted. The router will automatically choose the best position. For optimum handling of pick-up ports, the following functionality in an ALM tool will be helpful:

- Ability to generate pick-up locations.
- Guarantee of a minimum of one pick-up position.
- Ability to extend the number of possible positions to a maximum.

### 4.4.5 Checking and evaluating an entire library

Checking and evaluating a large library of more than 1,000 cells can be a major effort. Spending only 10 minutes on each cell would result in a 160-
hour task. Clearly, this task should be automated. A library check and evaluation tool must perform the following checks:

- Common cell height for all cells.
- Design rules inside the cell.
- Transistor sizing and netlist.
- No design rule violations when cells are abutted against each other.
- Ports are on grid.
- Power busses are aligned and on grid.
- Ports are labeled and labels are in the correct position.
- Cell origin is in the correct position.

4.5 Modifying Large Databases

To support large databases of 1 GByte and more, special functionality for data handling is needed. Today, only a few tools can handle databases this large. The main problem is that CPU performance and memory is growing on a linear path, but the run time of the algorithm grows exponentially with the data, especially when dealing with layout modification algorithms.

4.5.1 Using distributed processing

The first approach to reducing algorithm complexity is to divide large blocks into smaller sub-blocks, which can then be distributed to different databases, disks, and hosts. To reduce computing time, the different sub-blocks can be executed in parallel on different host machines.

![Figure 4-25. Executing different sub-blocks in parallel on different hosts on a network by tiling the layout into data blocks.](image)
Distribution to several machines decomposes the problem into smaller sub-problems. Even if the total complexity of the problem is exponential, dividing it into many sub-problems brings it back to a linear scale, provided that the problem can be divided at all. Figure 4-25 illustrates the approach of tiling the layout into data blocks, which can be processed on a single CPU and then distributed to different host machines on the network. One problem which must be taken care of is the synchronization of the wires connecting the different blocks. This problem is equivalent to the problem of solving the simplest form of a two-level hierarchy.

4.5.2 Modifying hierarchical layouts

Hierarchical layout modification and compaction are among the biggest challenges in automatic layout modification. Even manually changing a hierarchical layout is very prone to error.

But what is meant by hierarchical modification? The most common understanding of this term is that the layout hierarchy is maintained during the layout modification. This means that during the manipulation no device, such as a transistor or capacitor, will move from one hierarchical level to another or from one cell to another. In addition, the number of cells should remain the same. No cell should be renamed or duplicated during this process, because this is an implicit explosion of the data.

Some layout manipulation and compaction tools cannot support this definition of hierarchy handling. Some tools create their own hierarchy that differs from the original one. This may increase the speed of the tool, but it is not compatible either with design flow, or with the designer’s desire to maintain data integrity. Other tools require very strict definitions of the hierarchical topology, such as exactly abutting cells and only one or two levels of hierarchy. The more sophisticated tools can handle any type of hierarchy including overlapping cells, many levels of hierarchy, and an arbitrary large layout.

For strongly hierarchical structures, such as memory and datapath, a multi-level hierarchy is required, and the cells usually must abut exactly in order to match ports. Some tools manipulate each cell individually, and then try to abut them. This results in a non-optimum area, because the tools must consider the worst case. Advanced ALM tools use a better approach: they modify the cells in parallel and dynamically update the port positions.

Some tools can only deal with very restricted hierarchical topologies. Usually, they do not allow overlapping cells and over-the-cell routing, because they manipulate the cells independently. When a cell has been modified by being placed in several locations and over-the-cell routing is used, or when cells overlap with other cells, all local contexts must be
considered whenever a polygon in the cell is changed. This task is very complex and requires a kind of holographic representation of the cell that contains all of the environments of all cell placements. By using sophisticated automatic hierarchical analysis methods, some ALM tools are capable of doing this.

4.5.3 Using a hierarchy analysis and extraction tool

A major hurdle for hierarchical layout manipulation is the fact that there is no documentation about the hierarchy aside from the layout itself. The relative placement of the cells, port positions, and overlays must explicitly be made known to a layout modification tool. This information is quite complex when the approach is not limited to standard cell designs, but must also consider memories, datapath, or entire microprocessor cores. Therefore, a hierarchy analysis and extraction tool in combination with ALM methodology is very handy. Such a tool extracts the placement and port information in such a way that it can be used later, after modification, to rebuild the entire block. An analysis tool should consider different layout styles, such as standard cell blocks, memories, datapath, and random logic.

4.6 Defining and Entering Design Rules

The largest amount of time designers spend directly interacting with verification, compaction, and layout modification tools is in setting up the tools, since the execution itself is usually batch oriented. During setup, a lot of time is spent on coding design rules. Coding time for a design-rule setup with a traditional DRC tool can take four weeks or more. Even the most modern tools require this setup time because they need to be compatible with older tools.

These long setup times can be avoided. A simple and fast entry can be achieved using predefined rules and macros that satisfy most of the commonly used rules in the average technology.

These can be supported by menu-driven wizards or the graphical entry of design rules.

4.7 Preparing and Analyzing Layouts

Physical mask layouts do not contain explicit information about devices. Instead, they contain masks, such as metal, poly, diffusion, and contacts. If a tool is to make a meaningful manipulation of a layout, it must know the location and electrical sizes of devices. DRC and LVS tools can perform this
kind of device extraction via Boolean mask operations. However, they do not make the link to the original polygon that may need to be changed.

When Boolean mask operations are part of a layout modification tool, device extraction can be done internally and the necessary information can be assigned directly to the specific layout polygons. This approach supports a more accurate interpretation of design rules and a device sizing functionality.

In addition, Boolean and related operations can be used to create new or additional masks when migrating a layout to a different process. Examples of such masks are a second well, local interconnect masks, and high-voltage or low-voltage labeling masks, as well as phase shift masks. These masks are often created first, and later modified and adjusted to conform with actual design rules.

A very powerful method is to use mask operations to assign properties to layout structures. These properties can then be used to control the layout modification flow. Examples of such uses are the implementation of “don’t-touch” areas, special design rules, and ECOs.

4.8 Extracting Netlists

Pure layout formats such as GDSII do not store netlist information. However, this information must be present for net-specific sizing and spacing operations. This is especially true for hierarchical manipulation, in which the net is not represented by a single polygon, but by several polygons on different layers which are part of different cells. Therefore, it is very useful when an implicit netlist extraction from the layout is provided. An automatic comparison and match with the schematic netlist would be even better.

This netlist information can be attached to the physical design and used for signal-specific sizing and implementing signal-specific design rules.
Chapter 5

Integrating Automatic Layout Modification and Physical Design Reuse into Existing Design Flows

It is very difficult to introduce a new design approach that does not work together with existing design methodologies. Therefore, a new approach should be compatible with the existing design methodology and design flow. In this case, compatibility means:

a) The new approach can be combined with the existing approaches. The new approach should be an evolution instead of a revolution. It should enhance the existing design technology, but not completely replace it.

b) The existing technology does not need to be modified in order to work with the new technology. The new design process should be incremental to the old design process.

c) The new design technology provides a two-way street. It should make the process more efficient, but should allow the user to employ the old design technology in parallel. This is especially important during the introduction phase, until everyone becomes familiar with the new design technology.

This chapter will discuss the typical ASIC design flow, as well as the integration of different applications of automatic layout modification technology and physical design reuse into this flow.
5.1 The Typical ASIC Design Flow

Figure 5-1 shows a typical simplified ASIC design flow. Starting from a product specification, an architectural description of the circuit is generated using VHDL or RTL coding of the circuit, as well as schematics in some full-custom design teams. These descriptions are simulated for functional verification. From the functional description a gate-level description is created using automatic synthesis. This gate-level description is so detailed that static timing analysis or timing simulation can be performed on it. For this purpose, predefined timing models for each gate are used. These timing models are based on statistical data extracted from previous or similar designs. After the design has passed the gate-level timing verification, the netlist is physically implemented using automatic place & route tools. After place & route, the exact wire load information, consisting of wire capacitance and resistance, is available. This information is extracted from the physical layout and is fed back into the gate-level netlist. This process is called back-annotation. The back-annotated netlist now represents the more accurate physical situation of the design and the results of the simulation present a more accurate view of the final result. This process is also called “timing sign off”.

Within this flow, there are several checkpoints for functional and timing correctness. When the circuit design fails these checkpoints, the designer must go back in the flow and modify the input and parameters of the synthesis and place & route steps. For most designs using process technologies of 0.35-micron design rules and above, the typical number of iterations in this flow was between three and 10.

The ASIC flow described works best when the following three conditions are met:

a) The delay time between gates caused by the RC of the wires is small compared to the delay time caused by the transistor gates.

b) Synthesis tools estimate the RC wire delay of the output of a gate, according to the number of input connections to other gates. This number is called fan-out of the gate. The synthesis approach assumes that the number of fan-out connections is a valid estimate for the RC wire delay.

c) One implicit assumption of the synthesis approach is that the clock signal is synchronously switching at the same moment everywhere on the chip. Of course, this cannot be physically achieved because of wire resistance. The delay of the clock signal between different electrical nodes in the design is called clock skew. To reduce clock skew to a minimum, clock trees are generated that distribute the clock signal evenly on the chip.
Therefore, the effective automatic generation of clock trees is needed to make synthesis work.

![Diagram](attachment:image.png)

*Figure 5-1. Typical simplified ASIC design flow.*

The first two assumptions are no longer valid in deep sub-micron technology because the delay of global wires can now easily exceed the gate delay. The value of the wire RC cannot be estimated by the length of the wire or by the fan-out of the gate because the major contributors to this value are cross-coupling and fringe capacitance, which depend on the signal level of neighboring wires and on the spacing between wires. Therefore, the number of iterations in the design flow has grown significantly.
Chapter 5

The increase in design iterations works against the trend of higher circuit complexity and shorter time to market. Therefore, methods of shortening the number of iterations must be explored. One method is to limit the RC of global wires. This can be achieved through limiting the complexity of the design by splitting it up into smaller portions, which will be combined on the next level. Partitioning the design as shown in Figure 5-2 will also allow the reuse of different portions in other designs. To introduce this solution into the design flow, floorplanning is needed. But this approach does not guarantee that all wires will be short, because there are still global wires between the sub-blocks. The length and capacity of these global wires are difficult to estimate. However, the number of critical connections is smaller and they can be corrected by driver insertion, wire sizing, or rearranging the floorplan.

![Diagram](image)

*Figure 5-2. Partitioning a design to enable reuse of a specific part.*

The next method is a better integration of logic synthesis and place & route. The idea here is to consider the results of place & route during synthesis in a manner which does not require a full place & route cycle. Some synthesis tools have begun embedding the placement of gates into their flow. The purpose of this placement is to achieve a better and more realistic estimate of signal length. This is a reasonable solution when it is not necessary to consider dynamic cross-coupling effects and when lateral and fringe capacitance are relatively low.

The third method is to implement an additional feedback loop during global and detailed routing, which can increase accuracy.

All approaches have in common the goal of reducing the number of feedback cycles and of avoiding a complex, three-dimensional, full-circuit extraction. It must be admitted that all solutions researched to date have been aimed at design flows based on automatic place & route. In full-custom designs, such as datapath, memory, and even analog circuits, designers face the same problems but there are no new automated solutions available.
5.2 Adding Physical Design Reuse Capability to the ASIC Design Flow

Design reuse is the ability to reuse pre-defined circuits in a design, with the purposes of saving design time and of reducing the risks associated with redesigning the circuit. The first issue to be addressed is the design level on which the pre-designed circuit can be reused: VHDL, RTL, gate-level netlist, or physical layout. The freedom of design implementation depends on the level of abstraction: the higher the level, the more flexibility is available to the designer. But higher levels are also accompanied by higher risk, as well as demanding more integration work and more detailed knowledge about the circuit. The lower the level of abstraction, the less risk is involved, and the less integration work and knowledge is required.

a) The VHDL level requires detailed knowledge of the design in order to make design decisions and do the verification. The entire synthesis and verification flow must be executed. Even when synthesis and place & route scripts are provided, verification still must be done. Designers using this level of design reuse should be able to do the design by themselves from scratch. This approach is always required when no hard IP or layout is available for the design.

b) RTL and gate-level descriptions are more detailed, but also require circuit knowledge to do simulation and verification. The savings in design time are very marginal compared to VHDL design reuse.

c) The physical level provides the highest return in design time savings and security. This level of design reuse is employed for larger cores such as microcontrollers, DSP cores, datapath, and memory. Most of the circuits in these applications require either special expertise or extensive layout to match the specifications. The disadvantage of physical design reuse is that the architecture and specification of the circuit cannot easily be changed.

From the viewpoint of risk and effort, physical design reuse should be the method of choice. The main problem is that many circuits are combined with other circuits during the synthesis phase, so that no explicit layout is available for the circuit of interest, as shown in Figure 5-3.

The other problem often cited is that it is difficult to change the process technology and make the timing work in the new process when starting with a physical design. ALM technology can be used to overcome these disadvantages.
5.3 The Layout Conversion Flow

The process of moving a physical layout design to another process technology is called layout conversion. During this process, designers face two challenges. The first challenge is to match the geometrical requirements of the new process technology, so that the design can be produced in the new process with sufficient yield. The second challenge is to match the electrical specification of the circuit with the electrical capabilities of the new technology.

In most cases of physical design reuse, designers have dealt only with the first challenge. They calculate the geometrical relation between the old process design rules and the new ones and apply this factor to the design. They then simulate or produce the new physical design and characterize it. If the new design meets the specification it is accepted; if not, the design is...
begun all over again. This brute force shrink approach (see Figure 5-4) has been used mainly for designs produced in the same fab at different feature sizes. Most of the process technologies developed by integrated device manufacturers have been designed to facilitate a linear shrink from one generation to the next.

![Flowchart](image)

*Figure 5-4. The "brute force" shrink approach to physical design reuse.*

### 5.3.1 The geometrical process relationship

The geometrical process relationship can be defined as the relationship between the new and old process design rules. For shrinkable processes, this is mainly a single factor that is applied to all structures in the layout. Usually, such a single factor does not exist between different wafer fabs and especially not in DSM processes, because the different design rules change by different factors. If such a single factor is applied and satisfies all design rules, it does not result in the smallest possible layout for that process technology. Design teams therefore use a more aggressive factor, and to achieve a design-rule correct design the layout must be locally adjusted and corrected. This correction can be done before or after the shrink, and often requires a correction of the design grid because structures may be off grid after applying an arbitrary shrink factor.

Many design teams use a semi-automated approach to correcting the design. They apply mask operations like the ones available in verification tools and fix the remaining errors manually. In many large designs and full-custom designs, the effort becomes so huge that automatic layout compaction and layout modification technology is used to perform these adjustments.
5.3.2 The electrical process relationship

The electrical process relationship defines the relative electrical performance of the processes to each other. This relationship is important, because the same transistor fabricated in two different processes may display a different behavior in each process. Aside from the geometrical design rules, designers must therefore consider the electrical parameters of the new process when doing a linear shrink. When a process technology is developed to be shrinkable, the technologists take responsibility for this capability. But, if a design should be converted to the process of a different fab, the circuit designer is responsible for adjusting the design to the electrical process parameters.

There are several ways of evaluating the relationship between different electrical capabilities. One way is to compare electrical parameters such as the resistance, capacitance, and inductance of the different materials and elements. Another way is to simulate a typical circuit in both technologies and compare them. The simplest circuit to use for this purpose is shown in Figure 5-5, an inverter or a driver driving a load represented by an RC element. The designer can validate and adjust the transistor sizing in the new technology by comparing the delay of the circuit in the old and new technologies. If the technologies are of similar performance, the W/L relation should be similar. But this is often not the case because of differences between the technologies, such as doping, oxide thickness, and other parameters. In some cases, the W/L of the transistor can be smaller to achieve the same performance, but in other cases it has to be increased. Often the ratio between N and P transistors must be modified to achieve symmetrical behavior.

![Figure 5-5. A simple circuit which can be simulated in old and new technologies.](image)

The electrical relationship between old and new transistors, capacitors, and resistors can be easily calculated and measured. It is a little more difficult to estimate the relationship between the RC of the wires in the old and new designs. Besides considering the typical values, the overall circuit size reduction and the changes in distance between, and width of, the wires must be considered.
The resistance $R$ of a wire can be calculated to $(l*r)/w$, where $l$ is the length of the wire, $w$ the width and $r$ the typical resistance value per square. When the change of the resistance $R_{\text{new}}/R_{\text{old}}$ is of interest, the result is:

$$R_{\text{new}}/R_{\text{old}} = \frac{(L_{\text{new}} * r_{\text{new}})/W_{\text{new}}}{(L_{\text{old}} * r_{\text{old}})/W_{\text{old}}}$$

The wire resistance is a function of overall circuit size, the change in wire width, and the change in the typical resistance values of the processes. This means that the wire resistance does not change when the wire undergoes a linear shrink and the typical resistance remains the same. However, the resistance can increase when the wires do not shrink linearly or the typical resistance value increases, as shown in Figure 5-6. The change in wire resistance has to be considered when doing layout migration.

The estimate of the change in capacitance is a bit more complicated. The total capacitance of any node is the sum of different partial capacitances. These different capacitances are vertical capacitance and fringe or lateral capacitance, as shown in Figure 5-7. Vertical capacitance occurs between different metal layers and the substrate. Fringe capacitance occurs between different wires on the same layer. The model of capacitance is that of two plates which are separated by a dielectric. The amount of $C$ can be calculated according to the formula $C = k (A/d)$, where $A$ is the area of the plates, $d$ is the distance between them, and $k$ is the dielectric constant. The value of the dielectric constant depends on the material. The amount of lateral capacitance between layers depends on the thickness of the isolation between the wafers and the overlapping area between the polygons. The amount of fringe capacitance depends on the perimeter and height of the wires, the distance to neighboring wires, and the filling between the wires.
Chapter 5

The estimate of capacitance change can be done by analyzing some typical layout situations using minimum and typical design values. Usually very simple situations, such as the ones shown in Figure 5-8, are used for this purpose, because they make the calculation very simple. From this analysis a number can be derived for the typical change in wire capacitance.

Figure 5-7. Total capacitance is the sum of vertical, lateral, and fringe capacitance.

Figure 5-8. Simple layout situation for estimating changes in capacitance.

5.4 Applying Automatic Layout Modification and Design Reuse Techniques to New Designs

Physical design reuse techniques are based on the ability to modify, manipulate, and automatically adjust physical designs, also known as ALM technology. ALM techniques can also be applied to new designs for speeding up layout design, design-rule correction, layout compaction, and design rule adjustment. Design-rule correction can be used to automatically correct design-rule violations in manually drawn layouts. Design-rule adjustment can be applied if the process design rules change after the
physical design is already completed, and layout compaction can reduce the overall layout size of a design.

DSM design rules often require a grid size below 10 nm. The effect is that dimensions of the structures become larger in terms of the number of grids. For example, the size of a contact in a 0.5-micron technology is about 600 nm and the design grid is 50 nm, which results in about 12 grid units. In a 0.18-micron technology, contact size is reduced to 220 nm with a 10-nm grid, which comes to 22 grid units (see Figure 5-9). For 0.15-micron technology, the contact size is 180 nm and the grid is 5 nm, which results in 36 grid units. Therefore, it becomes much more time consuming to draw an optimum layout even for a small cell such as a standard cell. The reason for the increased design time is that designers have to zoom more often to make an accurate placement of the object. This additional work can be prevented by constructing the layout using a larger grid and applying ALM technology to adjust the layout to the final dimensions. This will speed up the design of the circuit and will not compromise the quality, because the changes are done locally inside the cells and do not affect general functionality.

![Figure 5-9. Dimensions of structures increase in DSM design, such as a larger number of grids relative to layout structure.](image)

5.5 An Incremental Approach to Timing and Signal Integrity Design Improvements

The main problem in current design is the increase of the iterations in the design flow, shown in Figure 5-1. The number of iterations can be reduced when a more incremental approach to design improvements for solving
timing and signal integrity issues is found. “Incremental” means that the
design can be improved step by step, with each step creating an
improvement over the previous result. In this way, the design is gradually
moving closer to the target specification. Chapter 3 deals in detail with
timing and signal integrity problems and how to solve them by applying
ALM technology. The main principle is that the problems can be corrected
by applying either device sizing changes, or routing changes, or both to
specific signals in the layout. Usually, the total circuit or layout topology
need not be changed. Instead, changes in wire spacing and transistor sizing
can correct most problems, as shown in Figure 3-12. Most companies have
in place automatic verification tools that identify the critical wires and can
specify required changes. The remaining problem is to implement these
changes in the physical layout.

Figure 5-10 shows the modified design flow with a layout adjustment
step. The advantage of this flow is not only that the overall behavior of the
circuit is not changed, but also that this flow is incremental. Extraction,
back-annotation, and timing analysis can be done for specific signals. They
do not need to be redone for the entire circuit, thus saving a lot of design
time and shortening the feedback loop.

5.6 Optimizing Power Consumption and Yield Using
Automatic Layout Modification Techniques

In addition to timing adjustments, power consumption has become a
major issue in high-speed and mobile applications. Some power optimization
can be done at the architectural level and even by redesigning the embedded
software. However, the real savings can be made by adjusting the physical
layout. Power reduction can be achieved mainly by reducing the capacitance
of the electrical nodes. It does not matter whether the capacitance is created
by the size of the gate or by the wiring: the more often a node switches the
more power it uses, and the more power can be saved. Therefore, power
optimization should focus on high-speed signals. There are tools on the
market that can identify the critical nodes and calculate new transistor sizes
or even wire constraints. ALM technology can be used to implement these
changes into the physical layout.

Figure 5-11 shows a design flow with a step for power optimization. This
kind of optimization can be added without negatively impacting the
performance of a circuit; indeed, it can even result in speeding up the circuit.
This flow also shows that only incremental checks are needed, because the
modifications are done locally.

Another type of optimization which can be achieved with ALM
technology is yield optimization. Many fabs offer recommended design
rules, which should be used wherever the size of the circuit is not impacted. With ALM technology, these rules can be selectively implemented.

![Diagram of Modified ALM design flow with timing analysis step.](image-url)
Figure 5-11. Modified ALM design flow with power optimization step.
Chapter 6

Applying Physical Design Reuse to Different Design Types with Automatic Layout Modification Technology

Physical design reuse using ALM technology can be applied to several different design types, such as digital or analog, standard cell libraries, memory, or physical compilers for memory and datapath. Since many of these designs are created by specialists in those particular fields, different design methods are used. A design reuse method must therefore consider the different requirements for each design type. This is the main reason that ALM for design reuse is not usually applied to an entire SOC design at once. Instead, it is more appropriate to consider the different parts of an SOC design separately.

This chapter gives an overview of how to use ALM technology for physical design reuse of different design types and styles, including standard cell blocks; full-custom designs, including datapath and memory; analog circuits; standard cell libraries; and physical compilers. Chapters 9, 10 and 11 discuss design reuse flows for the different applications in detail, based on real projects.

6.1 Digital Macro Blocks

In the past, most physical design reuse was done with digital macro blocks, either automatically created standard cell blocks or custom-designed blocks. The motivation for design reuse was often to go to a different or smaller process technology to save costs, or to combine or expand existing
designs. Often the long design cycles for complex custom circuits like microprocessors demanded a design reuse approach.

### 6.1.1 Standard cell blocks

Standard cell blocks are automatically created using a design flow based on synthesis and automatic place & route. The principal flowchart of the design flow is given in Figure 6-1. The easiest and most common way of performing design reuse on these blocks is simply to re-synthesize and replace & route the block. This approach is very feasible as long as the speed of the design is not very high and the number of gates is not too large. In this case, the risk of new or additional timing errors will be low and the number of design flow iterations will be small. When designs are larger, or the circuit frequency is above 500 MHz, the design process is more complex: the verification effort is much greater and the number of design iterations increases. The designer must also be very familiar with the circuit and cannot rely simply on automatic tools. UDSM designs are usually highly complex, and require advanced circuit speeds. For these designs, modifying the pre-existing physical layout design is often a good reuse approach.

![Figure 6-1. Design flow used for creating standard cell blocks.](image)

The synthesis and place & route design flow is not always straightforward for high-frequency circuits. Often, many iterations are needed to achieve a valid solution. In the past, manual re-routing of critical wires in the final block was also done to achieve timing closure. After a certain amount of time has elapsed between the original design of the block and its reuse, the tool versions have changed and the new versions produce
slightly different results. In this situation the designer has to go through the entire design flow again. This is time consuming and risky. In these cases, designers will want to convert the physical layout, even when it has been automatically generated. This results in a shorter design flow and leads to more predictable results without many iterations. The reuse flow is shown in Figure 6-2. Since the conversion shown here will not modify the placement or the routing, the relative signal timing among signals stays the same. This approach thus needs fewer iterations, is straightforward and more predictable, and therefore less risky.

![Design flow for converting physical layout for design reuse.](image)

High turnover in the engineering job market over the last few years has meant that key employees who did the original design may now be working on different projects, or may have left the company altogether. In order to get the circuit re-synthesized, circuit knowhow must be transferred to a new design team. Such a design reuse project may need the same, or even greater, amounts of time and resources as did the original project. Therefore, many experts stress the point that a special design-for-reuse flow is needed. Such a
flow must implement the design according to general conventions and must ensure proper documentation to enable third parties to reuse predesigned circuits. However, this implies that people design especially for future reuse, which is normally not the case.

Applying design reuse techniques to the physical block can help solve design problems not only in a later reuse phase, but also by solving timing and power problems in the initial design. Transistor sizing and wire sizing capabilities can shorten the initial design flow, as well as shift the physical design toward a new application in a different SOC design.

Reusing a standard cell block’s physical design cannot solve the problem if the specifications and architecture of the circuit need major changes. In this case, starting from a modified VHDL description is appropriate, although doing so will not ensure major savings in design costs.

6.1.2 Digital full-custom designs

Central processing units (CPUs) and digital signal processors (DSPs) are often large and complex systems that can range in gate count from around 10,000 to several million. They are often embedded in applications that require high speed and/or low power consumption, as well as a low price. Because of these requirements, their layouts are often done in full-custom style: this means that the layout is drawn manually in a layout editor without using a pre-designed standard cell library. This approach leads to a long design cycle that can stretch over periods of several years. This long and costly design cycle means that these circuits have been subject to reuse in many applications over many years.

The preferred, and usually the only possible, method of reuse for these circuits has been the linear shrink method combined with manual correction of the remaining design-rule errors. Successfully applying a linear shrink entails meeting two main requirements.

a. Electrical compatibility between source and target process

The first requirement is that the source and target processes are somehow electrically compatible, so that the electrical parameters, as well as the design rules, will shrink in a similar, symmetrical way. For example, the relationship between the drive strength of N and P transistors should be similar in the two processes. In addition, the typical values of wire resistance and capacitance should be in line with circuit speed. This process compatibility has often been guaranteed by design, which means that a new next-generation process is designed in such a way that former designs can be shrunk and reused in that new process. This has been possible because CPUs and DSPs have been owned and produced by
companies that own their own production fabs. Without this process compatibility, design reuse using a linear shrink is nearly impossible.

b. A common shrink factor for geometrical design rules

The second requirement is that there must be a common shrink factor that translates the source design rules into the target design rules. This means that all metal pitches, poly pitches, and other design rules must change by a linear factor. However, sometimes it is necessary to do some biasing on the layers by doing oversizing or undersizing. In many cases, a common shrink factor has been part of the process design. Often, this factor is compromised and non-optimum factors are applied. This works as long as the overall performance of the circuit improves and the area is reduced, i.e., the circuit becomes cheaper to produce.

The linear shrink approach has lost a lot of steam in recent years. It has been challenged by the limitations of physics and by technical and economic changes in the industry.

6.1.2.1 Timing closure and PDR

A major challenge for designers in deep sub-micron technologies is the timing closure problem. The basic effect in DSM is that the traditional relationship between transistor switching delay and wire delay is now reversed. This problem is discussed in detail in Chapter 3. However, when a given design is reused in a smaller process technology the problem is not as big as designing a circuit from scratch. The reason is that the relative wire length does not change. Nevertheless, it is useful to verify timing after the conversion, because the parameters for vertical, lateral and fringe capacitance have changed. If verification shows that the circuit has timing problems after conversion, this can be taken into account during the next conversion by adjusting wire spacing, wire width and transistor sizing.

Since the process of timing verification is no different from the timing verification of a new design, this is not an additional step in the design flow. In the past, many designs were reused without doing an explicit timing verification, because the technical capability was not available in the design flow or the circuits were simply too complex. This was often the case when a design was shrunk linearly and the verification was done by testing the silicon wafer.

6.1.2.2 Early process access and decreased ramp-up time

Over the last 10 years, the pressure on process development teams has increased, because design teams demanded early access to process information like design rules and process parameters. Up to the early ninties,
about four to five years elapsed between introductions of each new technology into mainstream production of logic products. The reason was that the process development drivers were commodity products such as DRAMs, SRAMs, EPROMs, or other high-volume chips. They had a much lower design complexity than logic products, but were very demanding in terms of process technology. These commodity chips were also the cash cows of their manufacturers at that time. Therefore, new process technologies were developed first for these products and later adapted to logic products and microprocessors. By the time these processes were transferred to logic products, the technology had already advanced and linear shrinks were easily made. But in the era of SOC designs this approach is not possible because DRAMs, SRAMs, microprocessors, and many other blocks are now combined on a single chip. Thus, it no longer makes sense to develop technologies for a specific circuit family.

As each new technology has to be applied immediately to complex designs there is no longer time for a long ramp-up phase. Product designs require a larger time span than process development and therefore have to be started before the technology is available. Designers have to work with simulated design rules and target technology parameters. These simulations are good estimations but their parameters continue to change during the technology development process, creating additional risks and delays in the circuit design. However, if a company waits for stable technology data, it will miss critical market windows and need not even start the design at all. The dynamic of technology development therefore imposes big risks on companies which do complex full-custom designs.

6.1.2.3 Need for design portability

Pure-play foundries are companies that do not design ICs. They only provide IC wafer production as a service to other semiconductor companies. They are focused on process technology and not on IC design. The services of these “fabs” are used by fabless semiconductor companies, and increasingly by traditional semiconductor companies. The foundry business is very competitive and it is often necessary for a semiconductor company to produce a design at several different foundries.

This economic situation requires a design style that provides portability among foundries. The standard approaches to ensuring portability are:

1. Using a superset of design rules for the different foundries. This is possible when the technology is frozen and the foundries have already been selected.

2. Increasing the minimum design rules to reduce the risk of later design rule changes. This approach is usually used when a design is started with preliminary design rules.
Switching to an ASIC design style approach that has the advantage of automatic layout. However, this approach gives up the advantage of why full-custom layout design is done in the first place.

6.1.2.4 Integrating external IP into SOC designs

An SOC design contains many different components, as shown in Figure 6-3. These components require a variety of design styles and expertise, but this diverse expertise is often not available in a single company, site, or design team. In addition, because of their complexity, the components cannot be designed from scratch when the SOC design is started. The time frame necessary to do so would by far exceed a reasonable time to market and the product’s entire market cycle. It is not economically feasible to allow five years of design time for a product with a six-month market cycle.

The most straightforward solution is to use pre-designed blocks, called IP blocks. These IP blocks must be compatible with the target process technology and the specification of the design for which they are used. There are two forms of IP: soft IP and hard IP. Soft IP is a software description of the block in a hardware description language, which can be implemented using synthesis and place & route. The entire design flow must be executed for each soft IP block. Hard IP is provided as a physical layout. It has to be adjusted to the target process of the SOC. In the past, this was achieved by performing a linear shrink on the IP block. In UDSM technologies, this requires the more sophisticated ALM technology.

6.1.2.5 Moving to place & route-based design flows

Most companies and design teams have moved in the direction of ASIC design, applying place & route technology for layout design wherever possible. This works very well as long as design frequencies do not exceed 300 MHz. Above 300 MHz, the place & route approach runs into the timing
trap described in section 6.1.2.1 above. Therefore, a lot of research and investment has been focused on this problem. Meanwhile, to overcome the speed barrier, many companies have re-implemented the most demanding blocks in full-custom style.

6.1.3 Structured full-custom designs

Structured full-custom designs are very regular designs which are often part of microcontroller and CPU designs. Examples include memory, such as RAM, ROM, and EPROM; FPGAs; and datapath designs. All of these have in common the fact that the same structures are frequently repeated. Many of these designs also require high circuit speeds. The layout for these designs is therefore optimized using a custom layout approach.

6.1.3.1 Datapath designs

Datapath components are functional blocks that perform computing tasks. Typical datapath components are multipliers, dividers, adders, filters, and shifters. In many applications, several such components are combined in a datapath. The performance requirements of a datapath are highly dependent on its final application. A datapath in a real-time speech recognition application has different performance requirements than it would if used in a toy robot.

High-performance datapath components are usually designed in a manually structured custom layout style. This style’s traditional advantages are the smallest area, shortest wires, and highest speeds. But in deep sub-micron process technology, the additional effects of cross-coupling and dynamic capacitance between wires must be considered. These effects become dominant in the high-frequency range and can only be detected after the layout is finished and parasitic circuit elements have been extracted and simulated.

If the simulation results reveal that circuit and layout changes must be made to adjust or eliminate parasitic elements, the result is a very labor-intensive process due to the custom layout style. One way to solve this problem is by using datapath generator programs, but their maintenance is expensive. Another method is adjusting the layout with automatic layout modification technology. This approach can combine the advantages of the optimum, full-custom layout design style with fast turnaround time in the design cycle. It also makes the reuse of pre-designed datapath components much easier.
6.1.3.2 Memory

The vast number of transistors in any circuit today are primarily used for memory, such as DRAM, SRAM, ROM, EPROM, and cache memory. Besides differing in type, memory also differs in size, speed, and addressing modes. Several years ago, most memory was implemented on separate chips, but today most are integrated on-chip together with CPU, DSP, and graphics processors, or integrated into an SOC design.

Because memory comprises the largest portion of transistors in a circuit, these transistors are usually packed very densely using a manual custom layout style. Often, certain design rules can be violated inside the core memory cell allowing higher density. Memory organization is highly hierarchical, which helps to reduce file size and design time. In the past, manufacturers developed specialized memory processes to make the densest possible structures, which resulted in memory becoming the technology driver for new process technologies. Because of the trend to integrate memory with other functions on a chip, this is no longer possible.

Since memory is now an integrated part of most chips, an even greater number of flavors are evolving, and it has become a major factor in design schedules. Traditionally, the only established way to reuse memory blocks was the linear shrink combined with a lot of manual fixes. However, this has become less feasible with more complex design rules, and in situations where memory designs must be fabricated on several different foundry processes. Neither conventional layout nor symbolic compaction delivers the desired results when applied to the problem of memory conversion. Instead, a sophisticated ALM approach must be used, which can perform multi-level hierarchical compaction, substitute memory core cells, and adjust the electrically relevant dimensions of circuit elements.

6.2 Analog Circuits

Originally, there was a strict separation between analog and digital designs. Analog circuits were not integrated with digital circuits on the same chip. Most of the efforts in design automation and the increase in design complexity can be attributed to the progress in digital design. However, the real world is analog and therefore, at the border of every digital design there is some analog circuitry. In the simplest cases, these are I/O circuits.

With the introduction of increasingly complex ICs, or SOC designs, nearly every design has become a mixed-signal design. A single SOC design contains many analog functions, such as A/D and D/A converters, phase-locked loops (PLLs), and external ports. In many cases, the design and implementation of the analog circuit becomes the constraining factor in these chips’ time to market. The major factors in time-to-market delays caused by
Chapter 6

analog circuits are that the design is not automated, specialized knowledge is often required, and reusing predesigned circuits is difficult. Analog circuits are still designed at the gate and transistor level using schematics and SPICE netlists, and the layouts are primarily created in the custom manner using a layout editor. Even though many of these circuits are relatively small – usually not more than 1,000 transistors – they require several months of design time. In addition, several different blocks are often required in the design of a single circuit, which becomes a major cost factor. In addition to the problems of the manual design process, analog functions are much more sensitive to parasitic effects such as cross-coupling capacitance and wire resistance than digital functions. These parasitic elements must be measured and extracted after the circuit is implemented, and then corrected. This number of iterations can easily break the project schedule.

ALM technology is one of several that can be applied to automatically correct analog circuits at the layout level. In contrast to traditional compaction technology, ALM can consider wire spacing and width and does not compact the circuit to the minimum design rules. This is why traditional compaction technology cannot be applied to these circuits.

A more effective method of increasing the design efficiency of analog circuits is to reuse them. Because these designs are done in a custom style, the most efficient approach is to reuse the physical layout data. This cannot be done with the linear shrink method, because most of the device dimensions do not shrink in the same ratio as do the design rules. Nor can compaction be applied, because this smashes the layout to the minimum dimensions, which can destroy functionality. With ALM technology, however, both disadvantages can be avoided. The design rules and the device dimensions can be adjusted while maintaining the design’s functionality. After an initial port, the parasitic elements can be extracted and the design can be simulated. Simulation may result in the need for additional changes to electrical devices, which can be implemented in another conversion cycle or directly into the layout to achieve the final result.

6.3 Standard Cell Libraries

Standard cell libraries are the foundation of most digital design flows. Without them, no design flow based on synthesis and automatic place & route will work. Therefore, they must be available early in the design process. About three to six months is required to design and simulate a manually created library. Library development is based on process data such as design rules and electrical process parameters, so it can normally be started after the process is stable. However, process development often
Applying PDR to Different Design Types with ALM Technology

requires test designs in order to create this data. In addition, the intervals between new process versions have become shorter. Thus, library development time is now delaying the introduction of new chips in the latest technologies. Many compaction tools and layout synthesis tools are therefore focused on standard cell libraries.

The layout of standard cells must follow certain rules so that they can be used by automatic place & route tools. These rules are usually:

- All cells must have the same height.
- Cells must be design-rule correct when they are abutted against each other.
- The bounding box and the ports have to obey the routing grid.
- The cells must have a common power bus width.

An automatic layout approach must be able to implement these requirements automatically.

Designs in the latest technologies – with their huge transistor counts, varying voltage levels, and analog and digital circuits – often require a set of separate standard cell libraries for different applications, such as different voltage levels, speed requirements, and densities. Some projects need three to five distinct libraries. Often, not all of these libraries are available, so designers must work with only one or two libraries.

For the latest timing-driven place & route tools, even more flexible libraries are desirable. These tools require many more cells with different drive strengths to optimize timing. Ideally, these cells should be generated automatically according to the router’s request.

![Diagram](image)

*Figure 6-4. ALM technology can be used to create different types of standard cell libraries.*

Another way of reducing the design time for a standard cell library is to convert an existing library to a new process using ALM technology. This approach can also be used to create different types of libraries, as shown in
Figure 6-4. The turnaround time can be reduced to under four weeks without compromising the library’s quality.

6.4 Physical Compilers

Another method of boosting the design performance of regular structures, such as memory, datapath, and multipliers, is to apply generators or compilers. Compilers can generate the blocks needed for different sizes, applications and speeds. A compiler usually consists of two components: the placement engine and the library of leaf cells. The placement engine is a software program that places the leaf cells according to the user’s specification. If the compiler is to produce memory for a different technology, the library of leaf cells must be adjusted to the new process. In addition, a set of configurations which the compiler can generate must be simulated to generate timing information.

As processes change more often and manufacturers allow the fabrication of their designs in more than one fab, the task of migrating compilers to different technologies has become quite common. Migrating the leaf cell library requires that the migration process can consider the library’s requirements, and can make adjustments to device sizes, as well. All possible compiler configurations must be considered for a migration; otherwise, the new compiler may create errors or may fail for certain configurations.

Hierarchical ALM technology can help to make physical compilers technology-independent by migrating the leaf cell library. The migration process must consider all of the compiler’s configurations.
Chapter 7

Layout Guidelines for Physical Design Reuse and Automatic Layout Modification

As compaction, verification and other ALM tools have improved over the last few years, the number of input layout requirements have consequently diminished. Some tools continue to be more restrictive than others, mostly concerning hierarchical layouts and hierarchical layout structures.

In general, some overall layout design guidelines should always be followed even when automatic layout modification is not being applied. This chapter presents these general guidelines, as well as specific guidelines for hierarchical layout modification, partitioning layouts for physical design reuse, designing layouts for portability, and laying out analog structures.

7.1 General Layout Design Guidelines

The following general guidelines are usually considered good design practice. Even if they are not required by specific tools, using them makes life easier.

7.1.1 Laying out polygon structures

Polygon structures should always be free of self-intersecting polygons and acute angles, which are usually not allowed in the layout database and can easily cause problems for many tools. When it is necessary to use them for company logos or similar identification marks, they should be placed in a separate cell, so that they can be easily removed from the layout.
7.1.2 Positioning text

A good practice for handling text is to use a different text layer for each data layer. If a text layer is shared for several drawing layers, e.g., metal 1 and metal 2, then the text should be clearly placed on one polygon and not on a crossing of two wires. Such a wire crossing placement would make it difficult to assign the text to one of the signals when they are not connected. Figure 7-1 shows examples of good and bad text placement.

![Figure 7-1. Examples of good and bad text placement.](image)

7.1.3 Using diamond- and octagonal-shaped structures

The use of diamond- or octagonal-shaped contact structures was sometimes allowed in early process technologies, but not any longer in modern technologies, because they violate design rules. As shown in Figure 7-2, their use results in a width violation and a compaction tool would only make the contact much larger. Some tools can automatically recognize such structures and convert them into orthogonal ones.
7.1.4 Using 45-degree transistors

Transistors rotated 45 degrees, as shown in Figure 7-3, are forbidden in most technologies because they violate design rules. Layout modification tools which cannot handle 45-degree structures cannot handle such rotated structures at all. Even tools which can handle 45-degree transistors may have difficulty sizing them correctly. However, the type of structures shown in Figure 7-4 are correct and widely used structures, which should be easy to size for tools that can handle 45-degree structures.
Chapter 7

7.1.5    Maintaining one-to-one device and cell relationships

Each device should be assigned to one and only one cell. For example, two different cells, one containing the poly gate and the other containing the diffusion source (or drain area) should not be used to build a transistor gate. The same is true if the transistor is divided into two parts, by cutting across either gate width or gate length. These examples are shown in Figure 7-6.

Some layouts place additional poly or diffusion pieces on top of gates or other devices. These polygons are drawn on another hierarchy level, as shown in Figure 7-5. Such arrangements are difficult to extract and often prevent correct transistor sizing.

Figure 7-4. These structures should be easy to size by 45-degree-capable tools.

Figure 7-5. When layouts place additional poly or diffusion pieces on top of gates or other devices, these can prevent correct transistor sizing.
7.1.6 Separating analog structures

For several reasons, analog structures should always be kept separate from digital structures. First, they are often more sensitive to manual or automatic layout manipulation. Second, they may be more sensitive to parasitic devices, and therefore have to be simulated more accurately than digital circuits. Since analog devices often must be sized differently from
digital devices, analog structures such as capacitors may need to become larger even while digital transistors can shrink.

Analog devices such as poly, diffusion, and well resistors should be labeled and marked as shown in Figure 7-7, so they can be easily distinguished from wires on the same layer.

![Figure 7-7. Correctly marking analog devices distinguishes them from wires on the same layer.](image)

### 7.2 Modifying Hierarchical Layouts

Most limitations of layout manipulation tools are related to hierarchical layout structures. Only a few tools are able to handle layout hierarchy without such limitations and still guarantee that the output hierarchy will match the input hierarchy.

#### 7.2.1 Cell abutting requirements

Most tools with hierarchical capabilities have strict requirements for abutting cells, as shown in Figure 7-8. Often they allow only a two-level hierarchy, as in standard cell designs. But most full-custom designs, as well as most designs created several years ago, can have any type of hierarchy, or none at all. Therefore, most tools with hierarchical capabilities do not work at all with such designs.
Some designs fulfill the abutting cell requirement almost completely with a mostly abutting hierarchy, but have some cell borders that overlap slightly in order to maintain design rules, such as shared contacts or preventing opens in case the design is shrunk. Abutting those cells would require the use of half-contacts inside the cells, which would flag design-rule violations when individual cells are checked.

Figure 7-9 shows an abutting situation where two cells abut and the port connection is made by a 45-degree wire. The wire is cut at the cell frame, which results in two wires with acute angles. This should be prevented, since it is a design-rule violation in the cells itself, and causes problems in the abutting process of most layout modification tools.

![Abutting cells](image)

![Overlapping cells](image)

*Figure 7-8. Most hierarchical-capable tools have strict requirements for abutting cells. Advanced tools can handle overlapping cell structures.*

![Figure 7-9](image)

*Figure 7-9. This 45-degree crossing of a cell border results in a design-rule violation*
7.2.2 Enforcing array-type base structures

Some ALM approaches require an array-type base structure in the layout, as found in datapath or memory designs. In case such a structure is not available in the layout, it must be enforced as shown in Figure 7-10. Some tools provide support functions for creating such a structure and can reverse the process after the layout is processed, so that once again the input hierarchy is matched.

![Figure 7-10](image)

*Figure 7-10. Some hierarchical ALM tools require an array-type base structure which must be enforced if it is not available in the layout.*

7.2.3 Maintaining rotated and mirrored cells

Because of the nature of many layout manipulation tools it is difficult to maintain rotated and mirrored cells. Most tools must actually duplicate rotated and mirrored cells, which can mean that instead of only one cell, there are as many as eight cells in the final layout, as shown in Figure 7-11. Of course, this is not a preferred solution because eventually eight cells instead of one have to be simulated and verified. This can cause problems, especially in memory designs. If a tool cannot handle mirrored or rotated cells, the duplicated cells have to be substituted at the end of the process to maintain exactly one cell.
7.2.4 Handling deadlock situations

Some hierarchical structures contain inherent deadlock situations for layout manipulation and compaction tools. One of the very simplest cases is shown in Figure 7-12. If the lower part of cell A is modified, the upper part has to be modified in parallel, which can, in turn, affect the lower part via the interface to cell B. This classic conflict can be solved in two ways. Either cell A is split into two cells A1 and A2, or cell B is renamed B1 and B2. The preferred solution is to split cell A, because it can be merged back after the modification. Some tools detect the deadlocks and support automatic splitting and merging of cells.

7.2.5 Handling naming and text conventions

Text layers should match data layers, as described in section 7.1.2. In hierarchical layouts, the label text should be placed on the same hierarchy level as the data. For layout versus schematic (LVS) check purposes, and device and signal matching, cell names in the layout should match cell names in the schematic. Doing so makes a hierarchical LVS much easier. Some hierarchical LVS tools cannot match the cells without matched cell names, and have to flatten the database.
7.2.6 Matching netlist and layout hierarchies

Many hierarchical verification tools require a match between netlist and layout hierarchies. If these hierarchies do not match, then parts of the design must be exploded, which makes it impossible to check large databases. A tool which cannot maintain the exact hierarchy as given in the input design is therefore useless. In such cases, the design has to be partitioned into blocks small enough to be processed and verified flat. This process is quite time consuming because it is not entirely automated.

7.3 Partitioning Layouts (Floorplanning) for Physical Design Reuse

Most designs are not initially created with design reuse in mind. Even when this is the case, a few planning steps at the beginning of a design can make later reuse of specific parts fairly easy. The main step for physical design reuse is physical design partitioning, or floorplanning. This floorplanning step guarantees that logical function blocks will stay together in one physical block and will not be distributed on other blocks or melded together with other functionality.
7.3.1 Reusing standard cell blocks

Many standard cell designs are done in a sea-of-gates style, as shown in Figure 7-13. Designers cannot reuse a specific part of a larger standard cell design without rerouting it, because the functionality is distributed over the entire sea-of-gates, not isolated within a single block. This is the main reason why many standard cell-based designs do not provide much opportunity for physical design reuse.

\begin{figure}
\centering
\includegraphics[width=0.5\textwidth]{sea_of_gates.png}
\caption{Typical sea-of-gates design style.}
\end{figure}

A strategy for enabling the physical design reuse of standard cell blocks is to route the designs hierarchically. The functional blocks are first routed separately, and then routed together on a higher level. This method may waste some area, because the sub-blocks may not fit together exactly, but it can also make verification easier because the designer can verify the timing separately for each small block. In addition, the functional blocks can be easily reused in other designs. Such a design would look like that shown in Figure 7-14.

\begin{figure}
\centering
\includegraphics[width=0.5\textwidth]{hierarchical_routing.png}
\caption{Routing a standard cell design hierarchically is one strategy for enabling physical design reuse.}
\end{figure}
7.3.2 Reusing full-custom design blocks

Full-custom designs are the major target of physical design reuse methods. They are arranged in functional units, and often combine handcrafted blocks with automatically placed & routed blocks if the area and performance requirements allow this. Most of the time, the functional building blocks are developed and verified separately to allow concurrent engineering. These blocks are often used in multiple configurations of a single chip family or in totally different applications, making them the ideal objects for physical design reuse. Sometimes these blocks are sold or licensed to or from third parties, which often requires a technology migration.

7.3.3 Reusing mixed-signal designs

Mixed-signal designs contain analog and digital portions. For reusing or converting these designs, the analog and digital functions should be separated on the layout and schematic levels so that they can be easily simulated and manipulated separately. This is necessary because analog functions often require schematic and netlist changes. The layout sizes of analog functions often change in completely different ways from the layout sizes of digital functions because they often cannot function at the lowest voltage level of the technology and the ability to shrink these circuits is limited. Therefore, it is often useful to place them at the border of the die.

7.4 Designing Layouts for Portability

In the past, design teams planned for design shrinks, or for porting a design to different wafer fabs, by using “unified design rules.” These were a superset of design rules that was compatible with all of the fabs being considered for a migration. Nowadays, with rapid changes in technologies and changing economic requirements, this approach no longer works because the technology data is often not available at the start of a design. However, there are some rules that can prevent technology incompatibilities during design migration.

7.4.1 Using orthogonal versus non-orthogonal structures

Many design teams previously restricted their layout designs to orthogonal structures, because these were considered easier to port. However, new design issues in deep sub-micron technologies are outdating this practice. For example, orthogonal 90-degree transistor bends are no
longer allowed in UDSM design. Instead, 45-degree transistor bends have to be used, as shown in Figure 7-15. In the future, cross-coupling and wave reflection effects may restrict even 90-degree wire bends. To handle these issues, flexible tools are required which can convert one design style into the other.

![Figure 7-15. Orthogonal 90-degree transistor bends have been replaced in UDSM design by 45-degree bends.](image)

7.4.2 Using local interconnect

Some companies have introduced a local interconnect layer that connects poly and diffusion without the use of contacts, as shown in Figure 7-16. Normally, it is easy to add local interconnect to a design lacking it, even when doing so is not optimum regarding layout area and speed. However, it is usually very difficult to convert a design in the opposite direction, that is, from one with local interconnect to a technology without it. Attempting to do so is like porting a three-metal design to a two-metal process.

A design team that wants to port a design to different, as-yet undefined fabs should avoid the use of local interconnect wherever possible. The savings in redesign effort may well justify doing so.
7.4.3 Handling butted tap contacts and shared contacts

The rules that vary the most among process technologies are the substrate and well-tap contact rules and the manner in which diffusion is defined. Some processes have one diffusion layer plus some definition layers, while others use multiple diffusion layers for the definition. Some processes allow butted tap contacts, as shown in Figure 7-17. Others do not allow this, but require that transistor and tap diffusions must always be separated. For process conversion, it is possible to convert butted to non-butted contacts, but it is very difficult and often impossible to convert non-butted to butted contacts.

Shared contacts, as shown in Figure 7-18, are very uncommon. Therefore, if at all possible, designers should avoid using them.
7.4.4 Using buried contacts

Buried contacts connect the diffusion and poly layers. They have been used primarily in DRAM applications. These contacts avoid a metal bridge to connect poly and diffusion (see Figure 7-19). However, this connection is very difficult to implement in most process technologies and its use has reduced wafer yield. Therefore, it is rarely used. It is, in principle, possible to convert a buried contact to a non-buried connection using metal in between, but this requires more space in the final layout. Poly and diffusion are not usually allowed on top of each other except in transistor definition areas. For this reason, designers should avoid the use of buried contacts even when the process offers them.

![Shared Contact](image)

*Figure 7-18. Shared contacts, which are rare, should be avoided.*

![Buried Contacts](image)

*Figure 7-19. Buried contacts are difficult to implement in most process technologies and can reduce wafer yield.*

7.5 Laying Out Analog Structures

When analog functions are converted to a different technology, transistor sizing and netlist changes are normally required. However, the transistors are not all sized according to the same function; indeed, it can sometimes be
quite arbitrary. For example, two transistors of the same original size may be sized differently in the target technology, one larger and the other smaller. In digital circuits, most transistor gates have either a minimum gate length or a minimum gate width. Analog circuits often require a specific relation of W/L with non-minimum W and non-minimum L (see Figure 7-20). To be prepared for these types of changes, the transistor gates and device types should be labeled so that they can be easily identified. It is also good practice to place analog structures at the periphery of the layout, so that size changes and circuit substitutions can be easily done.

**Figure 7-20.** Digital and analog transistor sizing.
Chapter 8

Guide to Physical Design Reuse Tools: Uses and Functions

Physical design reuse is a process requiring a variety of tools. Many tools in the physical design reuse flow are commercially available from different vendors, so the user has choices. However, until now there has been no tool or tool suite available in the market which covers all of the needed functionality. Therefore, users have had to create their own flows by combining different tools.

This chapter discusses the uses and functions of tools that are needed in a design reuse flow. These include design analysis, design optimization, layout modification, and design verification tools. However, not all of them may be necessary for a specific application or design environment. No vendors are mentioned in this section because fluctuations in the industry could quickly make such references obsolete. A good source for vendors of these tools are the following websites:

- www.edac.org
- www.dacafe.com

8.1 Design Analysis Tools

Design analysis tools are used to extract design information from a physical design. These include physical extraction and critical signal extraction tools. They may be needed for different situations:
The original design was created for a different purpose, and the new application has different requirements for timing and power consumption.

Since the input design was produced in a less critical technology, or for a lower clock frequency, it was not considered to have any signal integrity problems.

8.1.1 Electrical (parasitic) extraction tools

Electrical extraction, or parasitic extraction, tools extract the electrical parameters of the design from the layout. This is necessary because not all of the electrical parameters can be considered during the synthesis or schematic capturing process. These parameters include the capacitance, resistance, and inductance caused by the wires. However, the length and distribution of the wires is not known at the circuit level. In earlier VLSI designs down to the 0.5-micron technology node, wiring did not need to be considered as an electrical element, and resistance was assumed to be negligible. But in today’s UDSM technologies, a significant amount of signal delay can be caused by wire capacitance and resistance. On specific signals this may account for 70% or more of the delay. Because of the increasing importance of wire delays, extraction methods have been refined at each technology node. Today, four main levels of accuracy are used. These are one-dimensional, two-dimensional, and three-dimensional methods, as well as the extraction of dynamic capacitance effects.

8.1.1.1 Extraction methods

The one-dimensional method considers the vertical capacitance between wire and substrate. Capacitance is approximated by the area of the metal polygons on the different metal layers representing the signal. Resistance is approximated by the number of squares of the metal material and the contact area. This method is very fast but only gives a rough estimate, which is not sufficient for most designs below the 0.5-micron technology node. This type of extraction can be done by most Boolean mask operation engines, like those used in DRC and LVS tools.

In addition to these extractions, the two-dimensional method also considers lateral capacitance to neighboring wires. In UDSM technologies, the wire profile is changing from a more or less flat wire to a thin, taller wire. This results in capacitance that is created not by the plates formed between the substrate and the metal layer, but between the vertical wire boundaries of neighboring wires. Capacitance between wires of the same length can vary a great deal, because it is highly dependent on the routing density and voltage levels of the surrounding signals. Thus, two-dimensional
extraction is much more complex than one-dimensional extraction, but is still feasible for whole-circuit extraction.

Three-dimensional extraction considers the lateral capacitance between neighboring wires and the fringe and vertical capacitance between the different levels of metallization, as shown in Figure 8-1. This level of accuracy is needed in UDSM technologies, since they require many metallization layers. Their production processes require a process step that planarizes wafers before the next metal layer is applied. This planarization ensures that wires run planar over the wafer and do not form vertical waves, which causes stress and can break wires or cause light reflection problems during processing. However, with planarization the vertical thickness of the isolation oxide varies. These variations in oxide thickness affects the capacitance of the signals. Therefore, three-dimensional extraction is needed to get exact values. Because of the CPU time and main memory requirements, three-dimensional extraction is very expensive. Therefore, it is usually applied only to clock signals and other critical signals.

![Figure 8-1](image)

**Figure 8-1.** Three-dimensional extraction considers capacitance between neighboring wires, as well as capacitance caused by different levels of metallization.

The fourth level of accuracy to be considered is the extraction of capacitance in a dynamic state. This type of capacitance is relevant in fast circuits produced in UDSM processes. When two data lines with opposite voltage levels change their values, more than twice the amount of time is needed for them to attain the target voltage level. This is because the voltage level between the capacitor plates, as well as the distance between the plates, is factored into the capacitance value. The assumption in Figure 8-2 is that signals al and a2 are at VDD level and signal S is at GND level. When signal S switches from GND to VDD, the potential between the signals al, a2 and S shrinks down to zero until S reaches VDD. This means that the capacitance between the two signals is also continuously shrinking, which increases the speed at which signal S is changing. When the signals al and a2 are switching at the same time as S in the opposite direction, the amount
of time that it takes signal S to reach the threshold level of VDD/2 increases, causing signal S to be delayed. This is because the difference in voltage potential diminishes faster when S approaches the IV level. In the first case, the delay is d1, and in the second case, it is d2.

![Diagram](image)

**Figure 8-2. Cross-coupling increases signal delays.**

Therefore, the relevant capacitance for timing cannot be calculated from the topology alone. In addition, the signal levels must be considered. These can only be determined with electrical simulation. To do this for a multimillion-gate design is not only impractical, it is impossible in today’s technology. Yet failures caused by dynamic timing effects can occur in high-end circuits.

### 8.1.1.2 Determining critical signals

It becomes clear, therefore, that a method is needed for determining which signals have to be extracted and how accurate they need to be. Before the extraction can begin, each signal must be assigned a timing budget, which can be generated from the logic simulation. This timing budget is the
delay which was assumed during the logic simulation, or a kind of worst-case scenario under which the circuit still works.

The first step is to apply the fastest extraction method to all signals. Using a worst-case scenario, the likely critical signals are determined. The critical signals are the ones that might exceed their timing budget under a worst-case scenario. The next most accurate level of extraction is then applied to these signals. This is repeated until the highest level of accuracy available is reached.

In most design flows today, the most accurate extraction methods are used only for clock signals and a small number of other critical signals when the timing analysis reports timing problems.

8.1.2 Critical signal extraction tools

Many critics of physical design reuse emphasize the fact that circuit timing may not be correct after layout modification or circuit migration. The results of physical extraction tools, which provide capacity, resistance and inductance values, are used to verify the timing and functionality of a circuit. Verification is usually done via a static or dynamic timing analysis. Another method that is more suitable for physical design reuse automatically compares electrical values and identifies critical and relative changes during the conversion process.

8.1.2.1 Static and dynamic timing analysis tools

Timing analysis results in either confirmation that the circuit is functional as implemented in the layout, or a list of signals that are too slow or too fast to achieve the desired functionality. To get these results, the extracted electrical parameters representing the real delay in the physical implementation are back-annotated into the logic netlist. Often, the electrical parameters are first transferred into a delay model or delay parameter, which is more suitable for a logic simulator.

Dynamic timing analysis is done by simulation. Although it is a CPU-intensive process, dynamic timing analysis is more accurate than static timing analysis.

In dynamic analysis or simulation, the only paths in the circuit to be considered are those which are relevant and which will occur in real operation. In static timing analysis, the delay of all theoretical paths are considered, even if they might never be used in real operation. However, static timing analysis is less compute intensive and the algorithms are linear with the circuit size. Therefore, static timing analysis is the preferred method for large circuits with multimillion gates.
Neither method is 100% automatic. The results of both methods must be reviewed and interpreted by the designer. If timing problems are detected, the designer has several choices for fixing the problem:

- Modify the global architecture of the circuit.
- Change the logic and re-synthesize the circuit.
- Find a better physical implementation by re-routing or re-layout of the circuit.
- Change the drive strength of the signals.
- Modify spacing and width of critical wires to influence capacitance and resistance.

Changing the circuit architecture and the VHDL code or schematic of the circuits should be necessary only in the early phases of the project when different architectures are being explored. Doing several routing iterations and changing drive strengths of signal drivers has become very common in DSM technologies. Some designs report more than 20 such iterations. The benefits of modifying the width and spacing of wires are explained in detail in Chapter 4. Until recently, it was avoided because it was a manual process, but the ever-increasing influence of routing capacitance will drive designers to consider it more frequently. Using ALM tools, this process can be made more automatic and yield fewer iterations in the routing phase.

8.1.2.2 Critical signal RC delay comparison tools

One timing verification approach that is very well-suited for physical design reuse is the comparison of signal delay times before and after conversion. The delay of all signals is calculated before the conversion, and is compared with the results after conversion. This can be done when the primary reason for migration is circuit area reduction, not speed improvement. If speed improvement is the goal, this has to be factored into the comparison.

Physical design reuse has conventionally been done by performing a linear shrink on the layout, and making minor modifications to it. However, this approach has become increasingly insecure with shrinking process geometries. Therefore, more flexible processes must be used to change a design’s layout geometry. The most obvious method is using layout modification or layout compaction tools. However, with more flexibility the electrical properties may change over a wider range. To verify that these methods do not violate the circuit timing over a wide range, an effective approach is to compare the effective loads of the source and target designs.

This can easily be implemented by comparing the outputs of the parasitic extraction tools for source and result layouts. There are different comparison
options. One is to compare the RC elements of each wire, and the other is to convert these into delay times and then compare them.

Each signal is assigned an RC or delay range based on the input technology. Signals that fall outside the pre-assigned range relative to the input will be flagged for further investigation by static or dynamic timing analysis.

8.1.2.3 Noise, crosstalk, and signal integrity analysis tools

High-speed signals in the latest technology can face signal integrity problems, in which neighboring wires induce an incorrect signal level. These induced signal levels can produce incorrect results, and therefore can result in dynamic circuit failures, also called crosstalk failures. Lately, some vendors have begun marketing simulators and analysis tools for detecting these errors. The probability of such errors occurring increases with longer parallel signals, higher signal frequencies, and lower voltage levels. The combination of the reduction in the latest technology’s voltage range from 5V to about 1.5V and below, with the increase in circuit speed, has increased these problems because of the reduction in threshold value.

Since simulation and identification of signal integrity problems is CPU-intensive, this can only be done for selected signals and circuits. Therefore, the problem is to also define a selection process that can reduce the amount of signals which must be analyzed. In many cases, this identification process does not take place until after silicon is produced and circuit failures are detected. However, with the increase of these effects, this is no longer acceptable for future designs and the identification of critical signals has to become a part of the standard design flow.

When these analyses are done, they report critical signal problems. It is up to the designer to solve these problems by modifying either the physical design or the logic design.

8.1.2.4 Static and dynamic power analysis tools

As clock frequencies have surpassed the 1-GHz mark, circuit dimensions have increased, and the number of metal layers has grown to seven or eight, heat dissipation has become a major problem in chip design. One approach is to reduce dissipation by choosing better materials and packages. Another is to prevent the creation of heat in the first place. Since heat production grows more or less linearly with power consumption, the natural approach to solving this problem is to reduce power consumption.

Another reason to perform a power analysis is to prevent metal migration and voltage drops. Metal migration is caused by high currents and over time can destroy the wires, while voltage drops can reduce signal speeds and induce logic errors.
Power analysis tools highlight situations in the layout where power consumption may overheat the circuit, cause voltage drops, or create metal migration problems.

The main task of a power analysis tool is to estimate the power consumption on every subnet of the power net. Power consumption is the sum of the products of switching capacitance and frequency. Switching capacitance consists of gate and wire capacitance.

Power analysis tools are available in two types: static analysis and dynamic analysis. Static analysis takes into account the average switching frequency and the actual amount of capacitance. This is not accurate, however, because the switching frequency of a single transistor, or node, depends not only on system speed, but also on the values or pattern fed into the circuit.

Dynamic analysis simulates circuit activity and uses “real” switching activity for the calculation. Dynamic analysis is more compute-intensive, but it can give a better picture of the power density in different parts of the circuit. However, actual power consumption may also depend on how the circuit is utilized by the final end-user. For example, the power consumption of a microprocessor depends heavily on the type of application programs running on the final system. Some applications use certain units of the CPU as floating point units more often than other applications. Therefore, both methods can only yield estimates.

8.2 Design Optimization

The purpose of performing power and timing analyses is to ensure that the circuit is working according to the specification. These tools tell the user if some part of the circuit does not work at all, or that a serious reliability problem may occur after a short while. Design optimization goes beyond design analysis. The goal is to improve the design’s performance and power consumption without compromising functionality, costs, yield or circuit size.

8.2.1 Timing optimization tools

Timing optimization tools work similarly to timing analysis tools. The difference between them is in the information being sought by each analysis.

Timing analysis looks for what part of the circuit must be improved to fulfill the target specification.

Timing optimization looks for where and how the circuit can be improved above the target specification without adding significant additional costs to the design. The timing requirements in the specification are raised
until the circuit fails; the timing is fixed only when the costs of doing so are affordable in relation to the benefits.

For example, a design specified for 40 MHz can be optimized for 50 MHz or more to increase the production yield of 40-MHz parts, or the faster parts can be sold at a higher price point.

The base algorithms inside timing optimization tools are the same as those used for timing analysis tools. Some tools can combine timing and power optimization because reducing gate strength and capacitance in parts of the circuit where it is not required can save power.

The output of timing optimization tools is a new circuit netlist with new transistor sizes. It would be even more effective if the spacing and width of wires were taken into account to modify wire RC delay. This can make a large difference in UDSM designs.

### 8.2.2 Power optimization tools

Low power consumption is an ever-growing objective of today’s designs. Portable consumer applications demand power-saving circuits to increase battery life. High-speed computing is limited by the ability to handle heat dissipation. The task of power optimization tools is to indicate areas in the circuit that are most likely to provide the greatest power savings.

In CMOS designs, there are two main causes for power consumption:

- leakage current and
- switching current

Leakage current occurs on the device level and is inherent in the production process. It can be influenced in circuit design by using non-minimum gate length transistors or special low-leakage transistors. However, because these transistors are bigger and slower than minimum gate-length transistors, they are not usually used.

Switching current is always generated when transistors are switched. The transistor gate itself functions as a capacitor, as do the wires connected to the transistors. Every time a transistor is switched the capacitors are charged or discharged, which causes a current flow. The total power consumption in a circuit can be calculated by adding up the capacitances and multiplying that sum by the frequency. According to this calculation, there are three ways to reduce power consumption:

- Reduce leakage current
- Reduce frequency of transistor switching
- Reduce capacitance of wires and gates
The leakage current of a transistor can be reduced by increasing the transistor’s gate length. Some process technologies also offer special low-leakage devices. These low-leakage devices will only be used when timing requirements are not compromised. Reducing transistor switching frequency means eliminating redundant transistor switching. This can be done on the VHDL, RTL, and netlist levels.

Reducing the capacitance of transistor gates and wires, or identifying possible low-leakage transistors, can usually be done only after the physical design is implemented. The task of the optimization tool is to identify transistors that have a larger drive strength or transistor width than is needed to satisfy the timing requirement. Power consumption can be reduced by reducing the leakage current, and/or reducing the gate width to reduce gate capacitance. Therefore, these tools are often combined with timing analysis tools. Currently available power optimization tools focus on transistor sizing and do not provide specific wire spacing and sizing as output for the designer. This is because the input and output of power analysis tools is normally a transistor netlist, which does not carry wire sizing information.

In the future, these power optimization tools must also consider wire spacing and width, because most of the capacitance in UDSM circuits is now hidden in the wires, not in the gates.

8.3 Layout Modification

Before existing designs can be reused they must often be changed, since normally they were not originally designed for the new application. These changes can range from a total redesign and regeneration of the physical design to small manual layout modifications of specific devices. The main methodologies for layout modification are place & route, layout editing, compaction, and automatic layout modification. Automatic layout modification can include automatic layout editing, compaction, and place & route.

8.3.1 Place & route tools

Place & route tools are available from several vendors for different tasks such as gate arrays, standard cell blocks, and digital and analog applications. Most of these tools combine a set of different algorithms, breaking up the task into different sub-tasks. The algorithms include solutions for placement, global routing, and local routing.

The biggest group of P&R tools are designed for standard cell blocks. Gate array routers are becoming less important as the relative advantage of gate arrays in processing time and costs is diminishing, caused by the many
levels of routing layers (five to eight) used in today’s designs, versus the two layers used in early technologies.

Special analog routers also consider electrical characteristics of the wires, such as resistance and capacitance, during routing. So it is possible to route electrically matching or symmetrical nets, as well as a net with a specific resistance and capacitance. However, these routers can usually handle only a limited number of nets and are often not fully automated, forcing designers to interact directly with the system.

The trend in standard cell routers is to integrate them even more with synthesis tools to achieve timing closure for larger circuits with higher speeds. The problem that P&R tools face in UDSM technologies is that the wire delay becomes an increasingly larger part of the total signal delay. As explained in detail in Chapter 3, this is because as gate lengths decrease, circuits do not become smaller in absolute size. Instead, they have become larger over the last few technology generations while wires have become thinner, so that wire delay is a limiting factor in circuit design using P&R tools. The industry is trying to extend the usefulness of this approach by making changes in the semiconductor production process, as well as in the design process. In the production process, manufacturers are trading materials such as aluminium for materials with lower typical resistance and capacitance. In design methodology, companies want to integrate synthesis and P&R into a single tool or partition large designs into smaller portions with fewer design restrictions.

However, most design houses have no alternative aside from using P&R tools because this is still the fastest solution to getting products to market. P&R tools also play a big role in design reuse strategies for soft and hard IP. They are essential for the design implementation of soft IP. For hard IP reuse, they are used to combine reused macro blocks and cores, along with new design functionality, together on a single chip.

### 8.3.2 Layout editing tools

Layout editing is traditionally the most flexible, and the most time-consuming, way of modifying physical layouts. There are many layout editors on the market and some companies use their own proprietary tools. They can be distinguished by operating system, database management, database format, and capacity, and by whether groups can work on the same database simultaneously, or only one person at a time.

All layout editors have largely the same basic functionality. The differences often boil down to how user-friendly a particular editor’s interface is, and whether individual users are accustomed to it. Some editors
are integrated into a larger design system or framework. However, certain basic concepts should be considered when choosing an editor.

1. **Open database and external interface.**
   Does the editor provide an external interface to communicate with the outside world, aside from GDSII or GIF? Does it allow designers to export and import any data they want, aside from data which can be streamed out using GDSII or GIF? These features are important when designers want to write their own interfaces to other external or internal tools, and they can help to integrate the editor with other tools in the design flow.

2. **Scripts and macros.**
   Some editors allow designers to write or program their own scripts and macros for frequently repeated tasks. These scripts should be easy to write without a lot of programming interfaces, and should be integrated into the normal command structure of the editor so they can be easily accessed.

3. **Database management.**
   Editors or design systems which will be used by groups of designers working on the same database need database management for the layout database. This is necessary to prevent different users from modifying the same data and overwriting each other’s modifications. Such a system has resource control system functionality, such as tracking changes, and the ability to explore different solutions and then merge them later on.

### 8.3.3 Compaction tools

Although compaction technology has existed for over 20 years, there are very few commercial solutions. In Appendix 1, compaction technology is discussed in detail.

However, this technology will become more commercially important in the future because of shorter process development cycles and non-linear changes in future process technologies. The solutions to some problems the industry is facing, such as optical proximity correction (OPC) and phase shift mask (PSM) creation, will ultimately require some kind of compaction or decompaction approach.

### 8.3.4 Automatic layout modification tools

ALM is often performed with the mask operation capabilities of verification tool suites, which are intended for extracting different devices or layout situations. However, these capabilities can also be used to generate new masks, to bias masks in order to generate additional mask layers for final tapeout, or for migrating layouts to another process technology.
For new and future process technologies, more complex operations are required on layers to check, prepare, and migrate layout designs. This is because new technologies have stricter requirements for layer density, slotting of wide metal, OPC, and phase shift masks.

Another major application for an automatic layout modification tool is the electrical and geometrical adjustment and correction of the design on the physical level. Up until now, most of the fixes and corrections have been made using manual layout editing capabilities. However, with design data now reaching the range of several hundred megabytes to a few gigabytes, the job becomes far too complex and expensive for manual operation. Some companies have thus begun to adjust existing layout processing tools for the task of layout correction. However, existing algorithms have inherent limitations. Some new approaches therefore provide specialised tool suites, created specifically for this application, which understand a design’s hierarchy, design rules, and electrical parameters. The required properties of a general ALM technology are discussed in detail in Chapter 4.

8.4 Design Verification

Design verification is the process of verifying and validating that the design meets the specification. This process will take place at all stages of the design. These stages are the logic or functional level, the layout mask level, and the electrical level. For physical design reuse, the main requirements are the mask level and electrical verification.

8.4.1 LVS and DRC tools

LVS and DRC are the basic physical checks that must be done for layout signoff. The LVS check ensures that the new layout accurately represents the netlist, and that during the migration or reuse process no shorts, opens, or device size discrepancies have been created. The DRC checks that all physical design rules are met. For the latest technologies, complex design rules such as antenna rules, density rules, and signal integrity rules must be checked.

Several commercial DRC and LVS tools and packages are available. They differ in processing speed, how data is handled (flat or hierarchical), incremental checking ability, ease of use, flexibility, and error reporting and tracking.

1. Processing speed and data handling.

High processing speed is a major requirement of today’s multimillion-gate designs. The way in which the data is processed significantly affects
speed. Older tools only process a layout as flat data, which is not acceptable in high-end designs. Hierarchical tools that process each cell only once are much more effective. Even with hierarchical processing, it is often necessary to distribute large designs over several machines in a network in order to share their memory and increase compute speed. Tools optimized for an ASIC design flow can handle the hierarchy in standard cell designs much better than they can handle full custom layout designs. Some of these tools often flatten parts of the design that do not fit their specifications of hierarchy.

2. Incremental check.

Incremental checking can save a lot of processing time. With incremental checks, only that part of the design which was changed is reprocessed. In huge designs, this means that only a very small portion is reprocessed, thus taking only a very short time.

3. Ease of use, flexibility, and error location.

Ease of use has big implications for setting up the tool and for detecting errors in the design. Most tools use a textual description for design rules and commands, and then execute the check in batch mode. Implementing a new technology in such a system often requires several hundred lines of programming code in the tool-specific language, which can take several weeks or months. For most companies, this is increasingly unacceptable because of the subsequent delays in introducing a new process technology. Newer systems provide a GUI for defining the technology and an interactive step-by-step mode for testing the setup. These can reduce setup time to a few days and provide a much higher degree of correctness and safety. The major problem designers face using LVS tools is to find the exact location of errors. For example, when there is a short between VDD and GND the user often has a hard time locating it, because the physical size of the nets is huge. A hierarchical LVS check combined with cross-probe capabilities between layout and schematic can help to solve this problem.

The ease of use of LVS and DRC tools can be increased by adding automatic error correction to the flow. In some cases, over 90% of errors can be corrected automatically, which can increase productivity by a factor of 10.

8.4.2 Timing verification tools

Timing verification is a combination of parasitic extraction and timing simulation. To do timing verification, the tools and approaches described in Chapter 3, section 3.4, and Chapter 8, section 8.1 are used.
General Layout Modification Design Flow as Applied to the Alpha CPU Migration

The largest group of applications for layout modification technology consists of large macro blocks or whole chips. Examples are microprocessors, datapath or standard cell blocks. Therefore, these applications are a good starting point for explaining the general ALM design flow, leaving out the specific needs of standard cell libraries and memories, which are discussed in Chapters 10 and 11.

A typical example of such a project is the conversion of Compaq Computer Corporation’s Alpha CPU to the latest process technology. Therefore, much of this chapter describes the design flow of this specific project. In addition to serving as an example of ALM design flow, this project also contributes a lot of empirical data for physical design reuse.

The software used to implement the design flow of the project was the LADEE Tool Suite from Rubicad Corporation.

After a short overview of the general layout modification flow, the Alpha migration project is described in detail.

9.1 General Layout Modification Design Flow

The general layout modification design flow contains five main phases: linear shrink, database cleanup, hierarchy analysis and fractioning, correction, and layout reconstruction.
Chapter 9

The first phase is a hierarchical linear shrink, which is executed on the original hierarchical database. Care must be taken to ensure that no opens or shorts are created during the shrink.

The second phase consists of cleaning up the database. This is useful if the database is in bad shape, or if its size can be reduced to speed up later processing. However, this step is not needed for most designs, thus it is optional.

The third phase, hierarchical analysis, creates a topology database of the layout database to ensure correct reconstruction of the layout after correction. The analysis also fractions the layout into small leaf cells, which will be processed during correction.

The fourth phase, correction, is a complex flow containing different sub-tasks. This step does the hierarchical layout modification.

After correction, the layout is reconstructed in the fifth phase, using the topology database for reference.

9.2 Alpha Microprocessor Project Overview

Compaq’s third-generation Alpha microprocessor design, EV7, was completed in 2001 in a 0.175-micron bulk, 7-layer copper metal, local interconnect technology. EV7 was an aggressive design that used the earlier generation device as a processing core (EV7C), and added a high-speed memory interface, a high-speed glueless network connection, and a 1.75-MByte second-level cache (see Figure 9-1). The design outside the processing core was referred to as the EV7 periphery (EV7P). Processor pin bandwidth could achieve over 44 GBytes/sec. Connection to the system went through a 1,443-pin flip chip/land-grid-array package. The silicon interface to the package used a 250-micron pitch bump array with nearly 6,000 bump sites. The processor’s power requirement reached 155 W and
operation speeds were expected to exceed 1.2 GHz. There were some 153 million devices in the chip. To extend the lifetime of this design generation and to take advantage of ever-advancing technology, Compaq’s Alpha Design Group (ADG) set forth to recast EV7 into a smaller, faster technology, the EV79. The target technology was a 0.125-micron silicon-on-insulator (SOI) technology.

9.2.1 Migration methodology

The challenge for ADG’s EV79 team was to deliver that chip to market in a timely manner and address the daunting changes in technology. Design rules for the smaller technology were increasingly restrictive in some dimensions while more liberal in others. The relative footprints needed to be maintained so that assembly was not disrupted.

Alpha microprocessors can be characterized as custom, or structured custom. Most structures are hand built, or custom, with little reliance on re-use across functional sections. The concept of leaves or page cells across the project is not common. There are libraries of common circuits such as latches and some other primitives. As a result, circuit structures can be carefully tuned in layout using a lot of labor. Clearly, this style of construction is a handicap to the primary goal of the EV79 team, that is, time to market for the product. EV7, the source design for EV79, was built in exactly this fashion with about 4% of the design synthesized, and the rest in full-custom.

Complicating the shrink effort, the processing core for EV7 (EV7C) was a legacy design originally targeted for production in the Digital Equipment Corporation Semiconductor Fabrication Facility at 0.35-micron feature size and 4 layers of aluminum interconnect. That design was not expected to be moved to other fabrication facilities. Its first “shrink” was a smaller “in-house” process at 0.28 micron. Its second shrink was to the 0.175-micron process used by EV7 and built for Compaq by another manufacturer. Fabrication of EV79 would require the third rendering of the original physical core design in 0.125-micron SOI with a completely different metal stack, having rules very different from those of bulk.

In contrast, the EV7P design was begun in the 0.175-micron, 7-layer copper metal process, and had the advantage of not having to accommodate legacy structures. Most of the synthesis and reuse on the EV7 processor occurred in the EV7P. Furthermore, EV7 design started with an eye to migration, adapting many of the “recommended for migration” design rules suggested when the design started.

The contrasting design and execution styles presented two different challenges to the EV79 team. Resource and time constraints dictated two
paths: use Rubicad technology to convert the legacy EV7C and use skilled mask designers to convert the EV7P. Effectively, no logical changes would be applied to the EV7C, but only those physical and electrical changes required to adhere to the limits of the new technology. There would be no need for retrofitting new function or logic to any of the EV7C structures.

9.3 EV7 Microprocessor Core Database Assessment

The database of the Alpha core was divided into 20 sub-blocks. Some of the blocks shared the same library cells. The size of the blocks varied between 40 and 400 MBytes of hierarchical GDSII. In total, the 20 blocks comprised a database of over 2 GBytes of hierarchical GDSII. The total number of cells was over 18,000. Two thousand of these cells were used in two or more of the 20 blocks.

The different blocks of the database were designed so that, when placed inside the design, they did not create design-rule violations with neighboring blocks. This was necessary because most of the tools used in the design flow for editing and verifying the database cannot handle databases of 2 GBytes or larger.

Each of the 20 blocks contains up to 10 levels of hierarchy. This hierarchy has been modified over the years and became less structured over time because the design was reused often, as well as used in more than five process technology nodes and even more process options. In most cases, the hierarchy was not regular, and contained overlapping cells, polygons placed on the top, and even contact structures that combined several cells. Some blocks contained array structures, such as static RAM.

In addition, many parts of the design contained redundant polygons, which are polygons drawn on top of each other in different levels of hierarchy. Some blocks contained small polygons on the top level to fill opens caused by automatic gap filling needed in previous linear shrinks. In addition, some cells of the hierarchy were exploded, creating some very big cells.

The size and condition of the database would be a challenge for most verification and layout editing tools, and an even bigger challenge for a compaction and automatic layout modification tool. To date, no ALM tool had been able to cope with it successfully.

9.4 The Challenge: Layout Correction

As mentioned earlier, the Alpha layout database was used previously in process shrinks and reused for different applications. However, at this point the limit of shrinking the circuits without modification had been reached,
and the abilities of the existing design flow had run out of steam. Using the existing design flow, it would have required a huge amount of time and manpower to correct about 2.9 million design-rule errors in the database. With the available resources and using a manual correction approach, there was no feasible way to execute the project in a timely and cost-efficient way. Even if the financial resources were available, the project could not be completed in the available time frame of less than six months. An automatic approach for this task was therefore needed.

It was decided to shorten the time frame for the project without fundamentally changing the design flow by inserting an automatic design rule violation correction step, in other words, to automate as much as possible the manual work required. The LADEE Tool Suite was used for this added step.

### 9.4.1 Basic design-rule violations

There were several design-rule violations in the Alpha layout database which could not be corrected automatically with traditional mask operations. Figure 9-2 lists the main violations and the number of hierarchical error tick marks found for each one.

<table>
<thead>
<tr>
<th>Error type</th>
<th>Number of error marks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Poly Endcap</td>
<td>740566</td>
</tr>
<tr>
<td>Poly spacing</td>
<td>1494081</td>
</tr>
<tr>
<td>LI spacing</td>
<td>71073</td>
</tr>
<tr>
<td>LI cont overhang</td>
<td>221919</td>
</tr>
<tr>
<td>Contact spacing</td>
<td>109791</td>
</tr>
<tr>
<td>Gate spacing</td>
<td>255334</td>
</tr>
<tr>
<td>Total</td>
<td>2892764</td>
</tr>
</tbody>
</table>

*Figure 9-2: Main design-rule violations found in Alpha layout database.*

#### 9.4.1.1 Poly endcap

In order to do a linear shrink to the target process technology, the transistor endcap had to be enlarged by 25%. Normally, this could be done with mask operations, but in this case it would lead to poly spacing violations, as shown in Figure 9-3.

#### 9.4.1.2 Poly spacing

The poly spacing had to increase by 12.5% (see Figure 9-3). Therefore, the poly endcap could not be fixed automatically because doing so would also increase poly spacing violations. Spacing violations cannot be fixed by
mask operations because the wires have to be shifted by different amounts in different situations.

Since an increase in the absolute layout area had not been allowed for, a way needed to be found of compensating for the additional area. The new process allowed the use of local interconnect (LI), so an additional routing layer was available. Where no sufficient space was available, some space could be gained by converting poly wires to local interconnect.

9.4.1.3 Gate spacing

The rule for gate spacing of serial transistors increased by 11.11% (see Figure 9-3). Correcting these errors requires asymmetrically moving the transistor poly wires. The space needed to do this operation was available, because the contact-to-gate spacing, or local interconnect-to-gate spacing, could be reduced by half a drawing unit.

9.4.1.4 Local interconnect spacing

Local interconnect was inserted into the layout underneath the contacts. However, the spacing of these structures was violated and had to be corrected by increasing the spacing by 12.5%.

9.4.1.5 Contact spacing

The contact spacing for arrays of contacts increased. An array of contacts is defined as a field of 2 or more contacts by 2 or more contacts, as shown in Figure 9-4. Most of these arrays were diffusion power connections and diffusion input and output connections.
9.4.1.6 Contact/local interconnect overhang

The design rule for the overhang between contact and local interconnect distinguishes between situations with one contact versus ones with multiple contacts. The overhang for multiple contacts is three times smaller than that for a single contact in the specific target process. However, the multiple contact rule could only be applied when the contacts were placed at a minimum distance and belonged to the same electrical node (see Figure 9-5).

Inside the design, a lot of bar contacts were used to reduce the amount of data. These bar contacts were split automatically for final mask making and during the DRC. Unfortunately, the splitting algorithm distributed the contacts evenly, which created a wider than minimum spacing between them, so that the smaller overhang rule could not be applied. Therefore, the size of the bar contacts had to be modified to ensure that the splitting process produced contacts with minimum spacing.
9.4.2 Generating a high-threshold voltage mask

The target design required that some devices had to be changed to low-leakage devices. Low-leakage devices require high-threshold voltage and need an additional mask to identify them. This mask cannot be created by a mask operation alone, because of a conditional design rule. The special conditional design rule is that this mask must be either spaced at a certain distance or merged. Figure 9-6 shows a situation with some high-threshold devices.

![Figure 9-6. Correcting spacing violations by merging high-threshold mask polygons.](image-url)

The mask can be created by copying the active area of the specified transistors to the new high-threshold voltage layer and then oversizing it. However, after this operation there is a gap between high-threshold voltage polygons that is not large enough to cover the spacing rule. This gap can be closed, thus fixing the design rule violation. But this task cannot be done simply by mask operations using sizing operations or logic operations. This is because closing the gaps using mask operations may create new errors like width violations or spacing violations, as shown in Figure 9-7. To prevent these errors, edge moving operations have to be performed, which can be achieved with an ALM or automatic correction approach.
One major requirement of the Alpha migration project was to maintain the original layout hierarchy. There were several reasons for this requirement.

First, the size of the database did not allow further flattening without creating major problems for the various EDA tools in editing and processing it.

Second, parts of the circuit had to be simulated after the layout was corrected, but this would be impossible with a flat or mismatched hierarchy.

Third, the hierarchical verification flow might not work with a modified hierarchy. The amount of work required for further modification and correction of the remaining errors would have been multiplied on a flat database.

However, maintaining the hierarchy even with manual correction is often a challenge, because cells are placed in different contexts. If different cells are placed on each other, corrections in one place can cause problems in other places. After an error is corrected, the designer or the tool must check to make sure that no new violations in other placements of the cell are created.

Therefore, the correction process must take into account all instances simultaneously to prevent additional violations.
9.4.4 No metal modification allowed

To prevent additional signal integrity problems, the metal layers should not be modified in this project. This means that no layer should be modified above the contact layer. Of course, this guideline causes additional restrictions for the correction approach. However, when these metal layers did not need to be considered during correction, the advantage in this project was that the data size could eventually be reduced.

9.4.5 Maintaining the same number of leaf cells

The number of leaf cells should remain the same during the correction process. This means that even copying and renaming some leaf cells was not allowed, since doing so would have increased the data size and modified the layout hierarchy so that schematic cell names and layout cell names would no longer be the same.

9.4.6 Exclusion of certain cells and “don’t-touch” areas

The new SOI process required some new structures called “body ties,” which must be drawn in a very specific way. These body tie cells, along with others such as memory cells and some capacitor cells, should not be modified at all. Therefore, “don’t-touch” areas must be applied to these structures.

9.5 The Alpha Layout Modification Flow

The general layout modification flow was introduced in section 9.1. The Alpha flow differs from the general flow in that the first phase, linear shrink, was obsolete, and therefore did not need to be used.

Figure 9-8 gives an overview of the design flow showing the remaining four phases: cleaning up the database, database analysis, layout correction, and layout reconstruction.
9.6 Phase 1: Hierarchical Linear Shrink and Lambda Approach

The hierarchical linear shrink multiplies all coordinates of the layout structures with a user-defined shrink factor. The coordinates then must be snapped to the drawing grid. If snapping polygon edges and cell reference coordinates is done blindly without considering their specific contexts, opens or shorts can be created. This problem has been discussed in more detail in Appendix 1. To ensure correct grid snapping, a detailed automatic analysis of the layout connectivity is needed for this step to maintain the netlist integrity of the layout.

A linear shrink is needed when the layout is drawn in wafer dimensions. This means the drawn layout structures have the same dimensions as the final structures on the wafer. However, many designs with a long design cycle which are used over several technology generations use a different
approach called “lambda.” This approach does not require a linear shrink. Instead, the design is drawn in fixed drawing units which are scaled to the final process parameters by changing the value of the unit, for example, the gate length is drawn as ten units. For a 0.35-micron process, the unit is set to 0.035 micron to get to the right gate length. To convert this layout to 0.25 micron technology, the value of the unit is changed to 0.025 micron. This approach requires that the ratio between the design rules stays the same and does not change from process to process. If it does change, the original layout has to be adjusted.

The Alpha database was drawn using the lambda approach, so a linear shrink step was not needed. However, the layout did have to be corrected, because the ratio between the design rules had changed significantly in the new process.

9.7 Phase 2: Cleaning up the Database

The purpose of cleaning up the database was to reduce its size wherever possible, so that the resulting smaller database would run faster through the entire conversion process. This was done by removing redundant polygons, introducing additional levels of hierarchy, or removing unconnected material, as appropriate.

9.7.1 Removing redundant polygons

The layout contains situations where polygons are duplicated on a higher level of hierarchy. If left in the database, these polygons would significantly increase the database size and make it much more difficult to handle during the subsequent conversion. In addition, the presence of this redundant data would require much more time for running DRC and LVS checks. Figure 9-9 shows an example with a redundant polygon, and again after its deletion. The process of deleting these polygons is not trivial, as the example in Figure 9-10 makes clear. Figure 9-10 contains three cells. Cell A and cell B are two logic cells. On top of each of these, cell C is placed. For purposes of clarification, cell C contains only one polygon. This polygon is redundant when placed over cell A, but not when placed over cell B. In this case, the polygon in cell C cannot be deleted. Of course, this is a very simple example. In a complex design it is not always easy to decide if a polygon is redundant.

The LADEE Tool Suite contains a function that can automatically identify the redundant polygons and eliminate them. This was done on the entire database, which significantly reduced the complexity of some cells.
Some array structures contained polygons on the top level placed over each instance. These were the results of previous automatic processing when earlier linear shrinks were performed on the database. The linear shrink created opens caused by the grid snapping operations. These opens were filled by little polygons on top of the cells. The cells containing these exploded polygons became very complex for processing. A better solution, which was applied in this case, is to reconstruct the hierarchy by placing the polygons in the leaf cells. That way, no over-the-cell data is created. Figure 9-11 shows an example of such an array structure.
9.7.3 Removing unconnected diffusion and local interconnect

The target SOI process does not require well ties and substrate contacts. Therefore, the substrate diffusion and well tie diffusion, together with the local interconnect and contacts, could be deleted. In some cases, where the local interconnect also connected other circuits and devices, the polygons were not redundant and could not be deleted. When the related polygons are drawn in different cells on different hierarchy levels, as they were in this case, it is even more difficult to analyze them. Rubicad used the LADEE Tool Suite and the Hyper-Cell approach to extract and delete the unconnected local interconnect and tap area.

9.8 Phase 3: Analyzing the Database

GDSII databases do not contain explicit information about the layout’s hierarchical topology and connectivity between cells, nor is this type of information provided by other means of documentation. Moreover, it is usually unknown to designers. However, this information is needed for the automatic correction and modification process and therefore it must be directly extracted from the GDSII database.

9.8.1 Extracting hierarchy information

The compaction, correction, and modification of hierarchical layout databases require two main types of information. These are topology or placement information and connectivity information. Topology information is the cell references on different hierarchy levels and the information related to the overlapping of cells and polygons on different hierarchy levels. The
connectivity information that must be extracted consists of the ports between cells and the port connections made by dropping polygons of metal and vias on top of the cells. All of this information must be stored in a database that can be accessed during the correction process. In the case of the Alpha processor, the LADEE Tool Suite’s capabilities were used to extract this information. Rubicad’s DECOR tool accessed the database to correct the layout.

9.8.2 Merging macro blocks

As mentioned earlier, the database was composed of 20 independent blocks with one common library. The goal was to maintain a single common library for the conversion, so that no cells were duplicated during the process. For this reason, the extracted databases containing the placement and connectivity information had to be combined into one big library. This meant that the constraints of all blocks had to be combined so that each cell would fit into the references of all blocks. Figure 9-12 displays this process.

In this case, the 20 blocks contained between 800 and 2,000 cells each. The combined library contained more than 18,000 cells. These cells were then converted in parallel on a network of servers.

![Figure 9-12. Independently analyzed blocks are merged into one common database.](image)

9.9 Phase 4: Layout Correction Flow

The layout correction itself is a complex flow, which was implemented using different tools of the LADEE Tool Suite. The first step was to extract
complex layout situations using the mask operation capabilities of AutoMask. The second step was to use DECOR to identify what the errors were and how they could be corrected, and the final step was to correct them. All cells had to be handled several times and many cells were processed in parallel to reduce runtime.

9.9.1 Preparing the layout for correction

Before correction can start the layout must be analyzed. This step is also called extraction. In the case of the Alpha processor, extraction included netlist extraction, device extraction, and the extraction of specific layout situations. These extractions were necessary because the design rules for the target process were context-sensitive. The process also required the generation of auxiliary mask layers which were used only for the correction process. This is actually not unusual, because the same technique is used when the layout is checked with traditional DRC tools.

9.9.1.1 Device extraction

Before correction can be done, the electrical devices inside the layout must be identified and located. This is necessary for both netlist extraction and layout correction. The correction needs device information because design rules are different for different devices, even when the same layers are involved. For example, the minimum poly width may be different if the poly is used for a simple wire, versus if it is used as a poly resistor or as a transistor gate.

Identifying devices can be done mostly by logical mask operations. For example, transistor gates can be extracted by an AND operation between the poly layer and diffusion layer. The extraction is more complicated with a hierarchical design. In that case, the poly polygon may be in a hierarchy level different from the level where the diffusion polygon is located. Although such a separation is not usually good design practice, some design situations arise where it is appropriate.

9.9.1.2 Netlist extraction

Netlist extraction is performed to identify the electrical nodes inside the layout. This is required when net-related conditional design rules are part of the technology. Examples of such rules are:

Rule 1: If same node, then use value A, else value B.

An example is shown in Figure 9-13. The situation shows a spacing rule, also called a notch rule.

Rule 2: Spacing metal > a if different node; spacing metal > b if same node.
Polygons or wires can often be merged into a single polygon when they belong to the same electrical node. If the correction tool can access the netlist information, it can eliminate the spacing between such polygons, and gain more space to correct other errors.

9.9.1.3 Extracting specific layout situations

Some rules require additional extraction of specific layout situations, because the rule values depend on the specific situation. Examples of such rules are end-of-line and conditional spacing.

a) End-of-line rules

End-of-line rules are asymmetrical overhang rules between metal and contact, or via, layers. Figure 9-14 shows an example of an end-of-line rule. In this case, two opposite sides of the via must have an overhang value larger than A when a value B is sufficient for the remaining sides. For the correction process, it is not only useful to know where these rules have to be applied, but also which direction is optimum for the final layout size. As the layout changes during the dynamic correction process, the direction of the optimum solution may change. The software used had the capability of dynamically changing the direction of the end-of-line rule, so that an optimum solution could be found that would not increase the layout size more than is necessary.

Figure 9-13 Net-specific and conditional design rules.

Figure 9-14 Examples of end-of-line rules.
b) Different poly-to-diffusion spacing

The design rules for the Alpha project required different spacing between the poly endcap and the adjacent diffusion. The size of the spacing depends on the width and length of the endcap. Some are shown in Figure 9-15. Each of these situations had to be extracted to find out if there was a design rule violation and if so, how to correct it. For correction purposes, the different situations were labeled in the database using properties which were attached to the edges.

![Diagram showing different poly-to-diffusion spacings]

**Figure 9-15.** Conditional spacing rule between poly and diffusion.

9.9.1.4 Merging polygons

The first step in preparing a database is to merge the connected polygons on the same layer. This is done to reduce the amount of data and to make it faster for the software tools. This step also converts self-intersecting polygons to areas, as shown in Figure A1-10 in the Appendix.

9.9.1.5 Combining polygons from different cell levels

The Alpha layout had been used for several generations of processors. It had therefore been frequently modified and contained a lot of unusual layout situations that were not acceptable in a “hierarchically clean” layout. One such situation is shown in Figure 9-16. It contains a poly contact with a poly overhang, which is part of two different cells in different hierarchy levels. This situation is confusing for the correction software. Therefore, it was extracted and the overhang replaced with a polygon covering the whole contact, instead of two different shapes abutting each other.
9.9.1.6 Enlarging endcaps

One of the major errors to correct was the 25% increase in the length of the endcap. This correction could not be done without creating additional poly spacing errors. Figure 9-17 shows some layout situations, although there are more options for fixing the poly spacing violations. One is to move the poly away, and when there is no space at all the possibility still exists of moving the poly to local interconnect. The move to LI created some additional space because some of the design rules are smaller. Figure 9-16 shows a situation where poly was automatically moved to LI.

Figure 9-16. Cleanup of layout hierarchy.

Figure 9-17. Correction of poly spacing violations with and without local interconnect layer.
9.9.1.7  **Enlarging end-of-line rules for LI over-contacts**

The process rules distinguished between single and multiple contacts. Multiple contacts had to be spaced with the minimum spacing value and belong to the same electrical node. For single contacts, larger end-of-line rules were defined than for multiple contacts. The implementation of this larger rule created spacing errors between LI and poly or diffusion. The spacing errors to poly could also be fixed by moving poly to LI.

9.9.1.8  **Combining cross-shaped LI on same node**

The process contained a minimum area rule for LI polygons. This rule was large compared to the requirement for the coverage of LI over-contacts. However, there were many situations where LI over-contacts could be combined. This is shown in Figure 9-18. The two LI crosses belonged to the same node. By combining two or more crosses into one polygon, the area requirement was met and the polygon could be reduced to make space to correct other errors.

![Design-rule violation](image)

*Figure 9-18. Merging of local interconnect on the same electrical node.*

9.9.1.9  **Creating pseudo-contacts for LI/poly and LI/diffusion**

The connection between LI and poly or diffusion is established by overlapping the two layers. No explicit contact layer is needed for the connection. To ensure connection, the area of the overlap must be of a certain size. However, the correction tool needed additional layers to identify the overlap between the layers, so additional contact layers for LI/poly and LI/diffusion were created. The rules for the size of the contacts were complex rules. The size of the overlapping area depends on the overlap and overhang on the two layers LI/poly or LI/diffusion. Some examples of correct implementations are shown in Figure 9-19. The specific layout situations were extracted and their properties were attached to the contacts.
9.9.2 Measuring design sizes

Design rules express the minimum or maximum values required for the structures used in the design. However, the motivation for the size of many structures is the electrical performance of the circuit. These sizes are the width and length of transistors, the width of wires, and the amount of contacts used to connect the different elements. A physical design reuse approach must distinguish between the structures regarding which can be drawn with minimum size and which require special sizing. Identifying the different situations requires measuring the size inside the input layout and calculating a new value for the target design. The calculation of the target value is controlled by user-defined sizing functions, determined by electrical behavior. The results of the transformation are stored with the individual structure and are considered during conversion and correction.

9.9.3 Checking and correction

Design correction can be split into two phases: checking, which locates the violations in the layout, and correcting these situations.

9.9.3.1 Checking the design

Most tools which check mask layouts for the purpose of doing a design rule check mark the errors, but do not also correct them. When a DRC is performed with the goal of correcting the layout automatically, more data must be gathered, and the layout must be back-annotated with the error information. This has to be done in the hierarchical context of the layout, because a cell can be correct in one instantiation and incorrect in another. In
addition, the error could be local, but to correct it the environment of the error might also need to be changed. Correcting an error can involve changing several levels of layout hierarchy. This can mean that even locally correct structures have to be changed to fix an error elsewhere. All these modifications have to be stored within the database for implementation during the correction phase.

9.9.3.2 Correcting the design

In the correction step, the final layout correction is performed. In this step, information about the errors stored as properties inside the layout is used to correct the layout, if possible, by moving polygons. During this correction step, only polygons belonging to the current cell are modified. Polygons belonging to other cells are considered during the correction, but are not modified. The reason is shown in Figure 9-20. Here, cell C is a neighbor of cell A and cell B. To correct cell A, the polygon in cell C has to be modified. If, during the correction of cell A, the polygon in cell C is modified, a new error in cell B would be created. Therefore, the polygon in cell C is only moved when cell C is corrected and the polygons of cell A and cell B can be considered. Thus, several iterations may be necessary to correct complex situations.

![Individual cells](image)

![Individual cells combined in various layout situations](image)

*Figure 9-20. DRC corrections have to consider all placements of a cell simultaneously.*

It is now clear that the process does not guarantee a 100% correct layout. Limitations similar to those in manual layout correction may arise. However, overall results show that 92 to 98% of all errors can be fixed automatically.
Figure 9-21 shows a cell first with initial error marks and then after the automatic correction. It is clear that the overall topology of the cell does not change, but all errors were corrected.

9.10 Phase 5: Reconstructing the Layout

At the end of the leaf cell correction, the whole layout has been reconstructed according to the original layout. This is an automatic step, which uses the information extracted during the analysis phase.

9.10.1 External verification

To ensure the quality of the results, an external verification step was performed consisting of both DRC and LVS checks. Because of the project’s constraints, a 100% DRC-clean layout could not be expected. Since it was not possible to change the metal layers during the correction, not all errors relating to those layers could be fixed. However, 94.2% DRC correctness was achieved and the layout was clean of wire shorts and opens.

9.11 Project Results

The project was executed in about three months. This time period included preparing the data, automatic correction, manually fixing the remaining violations, and final LVS and DRC runs. Data preparation included identification of high-threshold transistors, the insertion of body ties, and a pre-conversion LVS check. In order to use resources more efficiently, many blocks that did not use a large number of library cells which were also used by other blocks were handled independently. This allowed some blocks to be prepared while others were being converted, and the remaining errors of already converted blocks to be manually fixed, all in parallel. This significantly shortened the schedule.

9.11.1 Memory and CPU time requirements

After setup of the flow, it took one day or less per block to run through the automatic correction flow. Each block was assigned to between 1 and 4 CPUs to run in parallel. Ten x86-compatible 1.4-GHz machines were used, connected on a 100-Mbit local area network. Each machine was loaded with 0.5 to 1.5 GBytes of main memory and a 30-GByte hard drive. The total time elapsed was about 60 CPU days, although during this time not all CPUs were 100% busy. Some were idle because they had to wait for other processes to finish. Without any feedback loops between final check and correction, it would have taken about five minutes elapsed runtime per cell.
Using 10 CPUs, this would have added about one week to the total turnaround time.

Figure 9.21  Layout cell before and after correction.
9.11.2 Amount of error reduction

The average amount of error reduction was 94.2%. Some blocks reached 99% and some only 90%. A 100% correction was not expected, because it was not possible to alter the metal and contact layers. The total numbers were 2.9 million initial errors, which were reduced to 167,000 errors.

The manual correction of the remaining DRC errors took up to three person-days for the smaller blocks, and six to seven days for the larger blocks with a higher error count. Most of the remaining errors were easy to fix, because the automatic flow had already partly corrected them. The total manual effort was less than six person-months.
Chapter 10

Aspects of Memory Conversion Projects

Since memory is the most widely used type of macro block in IC design, it is an excellent target for design reuse. This chapter discusses the specific requirements of memory conversion and the best memory reuse strategies. It provides a general physical design reuse flow for memory and memory compilers and discusses specific problems in those areas.

Because every memory, memory compiler, and memory process is slightly different, this chapter does not focus on one specific memory architecture or process technology. The flow and information presented here is compiled from several projects. The problems, flow, and tasks described may not all be needed in one particular project. This chapter contains the memory-specific aspects of physical design reuse and layout modification. The more general aspects of reuse projects are covered in Chapter 9. The project flow given in this chapter is extracted from projects done by Rubicad or Rubicad’s customers using the LADEE Tool Suite.

Memory exists in different flavors for different applications. Some chips, like DRAM, SRAM, ROM, and Flash, contain only memory. These chips are designed in special process technologies and in full-custom layout style. Their structures are very regular and hierarchical, making the design effort relatively low compared to the effort required for SOCs. As technology features have become smaller and more devices can fit on one chip, designers have started to integrate memory with logic on the same chip.

Measuring by transistor count, the biggest part of an IC design today consists of embedded memory. Therefore, the design of embedded memory
has been automated wherever possible. However, because memory plays
such a major role in designs, a memory’s area and speed are critical. One
solution to fulfilling these requirements is the use of memory compilers for
embedded memory. A memory compiler is a software program that
generates or synthesizes memory for different size and speed requirements.
One of its elements is a library of technology-specific leaf cell layouts,
which are used to compose different memory configurations. These leaf cells
are usually created manually with a layout editor to achieve optimum results
in layout density and performance.

Each memory contains one or more special “bit cells” that store
information. They are usually composed of between one and six transistors.
However, they are referenced very often and thus occupy the biggest part of
a memory’s area. For example, a 1-MByte RAM contains about eight
million bit cells, representing eight million to 48 million transistors.
Depending on the architecture, the input and output driver cells are only
referenced a few thousand times. Therefore, memory bit cells are often
physically optimized by applying smaller design rules that consider OPC and
other effects of the production process. Most memory cells are therefore
defined by process technology engineers, not by circuit designers. These
cells are very technology-specific and change with each technology.

10.1 Typical Memory Structure

Figure 10-1 shows a typical single-port memory structure, in which nine
fields can be distinguished. The major fields are the memory bit cell plain,
the address decoder and word line driver, the sense amplifiers and output
drivers, and control logic. The other five fields contain power connections
and substrate contacts, which are very small and do not contain major
functionality.

In all memory over a certain size, the biggest part is the bit cell plain
containing all the bit cells. A small memory’s area can be dominated by the
size of the decoder and drivers. Therefore, many compilers have different
sets of leaf cells which are used for different memory sizes.

The main challenge in memory design is that, even when the decoder and
driver cells are much more complex than the bit cell, their pitch must match
the bit cell’s pitch. Therefore, these cells are usually very slim but also very
long. Usually, two or more drivers are designed in a row to match the pitch
of the bit cell. The requirements of smallest possible area, high performance,
and pitch matching of the cells for all possible memory configurations often
leads to multiple iterations in layout design, and therefore long design times.
10.2 Special Memory Conversion Requirements

The conversion of a memory, a memory library, or a memory compiler has different requirements than a macro block conversion. Following are the most common memory-specific conversion requirements.

10.2.1 Handling memory bit cells

A memory bit cell is usually provided as a special cell by technology engineers. The layout of each one is therefore different for each technology and each process node. Even when a memory is converted from one fab’s process to another’s, a new memory has to be substituted for the old one. One of the most difficult situations is when the technology node remains the same and only the fab changes. Here, the difference in size between the source and the target bit cell is very small. Therefore, a general shrink is not possible because afterward the design would be off-grid.

Another difficulty is that the metal wiring of the bit cell may not be exactly the same. For example, the order of wires is exchanged, or the metal levels for x and y routing are different. The layout must be adjusted for these differences: either the bit cell, or the interface between the bit cell array and the decoder or driver cells, has to be changed. Figure 10-2 shows the adjusted layout with a modified interface.
Chapter 10

10.2.2 Pitch-matching leaf cells

Another requirement is that the leaf cells have to pitch-match exactly according to the memory floorplan of the memory compiler. Otherwise, the different memory configurations cannot be automatically generated. This pitch-matching has to consider self-abutting, mirrored, and rotated cells. It may be necessary for some cells to match a number of different cells, not just one other cell exactly. Figure 10-3 shows different examples of pitch-matched cells.

*Figure 10-2. New memory bit cell requires that the routing of the decoder cell has to be adjusted.*
Aspects of Memory Conversion Projects

*Figure 10-3.* Example of pitch-matched cells.

*Figure 10-4.* Different memory areas have different sizing requirements.
10.2.3 Different sizing of different memory areas

The different areas of a memory have different requirements for device sizing. The bit cell of the bit array is pre-defined, and cannot be changed during conversion. Therefore, the shrink factor or area requirements may be different for each of these areas. To get optimum area results, this situation has to be considered during conversion. The degree of shrink may even be different in the x and y directions of the memory. For example, a memory cell grows by 5% in the x direction but reduces by 7% in the y direction. Or the output drivers may grow by 10% in the y direction and the row decoders increase by 5% in the y direction. All of these situations are shown in Figure 10-4. Of course, it is only possible to shrink each row and column by a single factor.

10.2.4 Considering all possible memory configurations

When several memories generated by the same generator are used in one ASIC design, they should use a common library. Otherwise, a lot of data must be duplicated. During the conversion of these memories, the common library should be maintained and instances should not be duplicated. Several memory configurations must be considered when converting a memory library which is part of a memory compiler. The selected configurations should represent all possible placement options of each leaf cell in any configuration. To achieve this, 10 to 20 different memory configurations are needed. The different configurations must be considered during the conversion, so that afterwards there is exactly one output cell for each input leaf cell. This means that the tool must be able to handle self-abutting, mirrored, and rotated cells.

10.2.5 Maintaining layout hierarchy for verification

Maintaining the exact layout hierarchy during a conversion is critical for several reasons.

a) Data size reduction.

Exploding the layout data of a memory into a flat database is unacceptable because of the data size. Several hundred kilobytes of hierarchical data can easily create hundreds of megabytes, or even gigabytes, of data when memory data is exploded.
b) Verifying the layout using DRC and LVS.

Since DRC and LVS tools cannot handle very large flat designs extending throughout several gigabytes of data, a hierarchical approach must be used. However, hierarchical tools require a very good correspondence between layout hierarchy and schematic or netlist hierarchy. This means that cells should be named the same in both layout and schematic and there should be no duplicated cells in the layout library from one schematic cell. When these requirements are not met, the tools may automatically explode the hierarchy, resulting in an unacceptably long runtime.

c) Electrical circuit simulation.

Even when the size of the memory still allows flat verification with DRC and LVS tools, the electrical circuit simulation will certainly fail. The reason is that the tools usually simulate a memory's worst case and assume that the cells are all the same. Under this assumption, only several hundred transistors must be extracted and simulated instead of millions of transistors. This results in a higher accuracy of circuit simulation and shorter verification times.

Clearly, maintaining hierarchy means maintaining an exact correspondence between layout and schematic hierarchies. Maintaining any hierarchy for the sake of having a hierarchical layout is simply not enough.

10.2.6 Special device sizing for drivers and sense amplifiers

The memory decoder, bit array, and output driver areas have different device sizing needs. The bit cell is usually given by the technology engineers. The size of the drivers inside the row decoders depends on the specific technology performance, desired access, and memory size. However, the size of the output drivers depends more on the context in which the memory is used. When the overall device is small and the wire load is small, the drivers can be small. For large designs with long wires at the output ports, larger drivers are preferred. These requirements must be considered during the conversion process.

10.2.7 Supporting a special grid for memory ports

Memory is usually used as a macro, abstract, or phantom in combination with the ASIC flow. For place & route of the layout phantoms, the ports have to match the grid of the routing library. The conversion tool must create new phantoms that fulfill this requirement.
10.2.8 Typical memory conversion flow

Figure 10-5 shows a typical flow used for memory conversion. It includes the main steps, and leaves out steps and details commonly used for the conversion of macro blocks or standard cells.

![Flowchart]

*Figure 10-5. Typical flow used for memory conversion.*

10.2.9 Creating typical configurations

The first step is to create all the necessary configurations that represent all possible contexts in which each leaf cell is used. These configurations are created by calling the memory compiler with the different configuration parameters. Detailed knowledge of the memory compiler is required to decide which configurations are needed, since exponentially too much data
would result if all theoretically possible memory configurations were created.

10.2.10 Reducing overall complexity

Some memory is very large and contains several million cell references to the bit cell and driver cells, when considered as flat data. However, when global wires such as power lines are crossing the bit array or the driver areas, a hierarchical tool needs a lot of runtime to consider all the instances. The analysis and conversion runtime for the memory can be significantly reduced by reducing the number of cell instances in the arrays. An example is shown in Figure 10-6.

*Figure 10-6.* Reducing the number of cell instances in the arrays can significantly reduce memory analysis and conversion runtime.
Chapter 10

The remaining cell placements have to represent all typical cell placements. Each context of a cell placement must be represented at least once in the remaining structure. Reducing the number of cell references is done manually and can take one to two hours for each memory. This is not a lot compared to a runtime reduction of days.

The missing references must be added after reconstructing the memories.

10.2.11 Analyzing memory configurations

The selected memory configurations have to be analyzed for cell placement and layout hierarchy. This step is automatically executed by the LADEE software. The result of the analysis is a lot of small files representing each library cell.

10.2.12 Calculating device sizing

Device sizing depends on several factors, which are the relative performance of source and target technology, and the target memory specification. The relative performance of source and target technology can be evaluated as described in Chapter 3. The target memory specification depends on the design’s overall speed, application type, and circuit size, as well as external factors such as the implementation of standards or compatibility with other products. Therefore, a general formula for calculating device sizing cannot be given. In most cases the specification requires that the memory’s performance should be about the same or better than it is in the input design.

10.2.13 Reduction or shrink factor analysis

Earlier, it was mentioned that different reduction or shrink factors must be applied to the different fields of a memory for optimum memory conversion. For example, in one application the new memory bit cell was provided by the fab. It was 4.3% larger in the x direction and 5.7% larger in the y direction. However, the transistor sizing did not change, since the gate length did not change, so the other driver cells did not change in size either. As a result, the schema for the area changes is as shown in Figure 10-7. Although shrink factors are the subject of this discussion, in this case it was a growth factor, because the new design was slightly larger.

The area estimate for the driver area was made by doing a trial conversion with the new design rules and device sizing on the relevant cells.
Conversion and correction flow

Conversion and correction is a flow containing several steps. The main steps are a linear shrink and automatic correction of the remaining design rule violations. The difference between this approach and a macro cell approach is that here, different shrink factors are used for different parts of the design. In one application, the shrink factors for the bit cell array had to be very accurate to match the size of the new bit cell, which was provided by the foundry. The different areas to be shrunk were defined graphically by drawing the grid over the layout, as shown in Figure 10-7. The shrink factors were also specified inside the layout, as shown. The tool picks up the information from the layout view, and the rest is 100% automatic. If there are conflicts, a report file is generated or the layout is modified. Conflicts can appear when a cell is placed in different parts of the memory or when cells are crossing the cut lines for the different areas. However, these conflicts are very unlikely in memory because the cells are usually used in only one area of the memory: either bit cell, driver, or control areas.

All other parts of the flow, including the correction and check of the final result, are the same as with macros or standard cell designs.

Figure 10-7. New memory bit cell from the fab is 4.3% larger in the x direction and 5.7% larger in the y direction. Since transistor sizing did not change, neither did the sizing of other driver areas.
10.2.15 Reconstructing the memory

Reconstructing the memory is done completely automatically. The tool uses the information extracted during the layout analysis to place the cells in the right locations. The memory design’s hierarchy is completely maintained. In some cases, the area utilization and flexibility of the tool are increased if very small cells like contact and via cells are temporarily exploded during the migration process. These cells will be added to the hierarchy at the end of the reconstruction process.

10.2.16 Creating phantoms for place & route

When specific memory configurations are converted to a new process, phantoms for the place & route tool must be created. The phantom’s requirements are that the memory’s ports have to be on the grid defined for the place & route tool. These ports are the address, data, control, and power lines. During conversion, the structures are arranged according to the technology grid, which is usually smaller than the routing grid. This is necessary for achieving the smallest area. To meet the requirements of the place & route tool, a frame is created around the memory macro and the interface signals are routed to pin positions on the frame. These positions must then be on the larger routing grid. This process is done automatically with each memory block.
Chapter 11

Aspects of Library Conversion Projects

Libraries are one of the main applications of ALM technology. As process evolution continues, fast access to libraries in the early stages of design is becoming a requirement for design success, especially for many young, fabless companies which cannot rely on off-the-shelf libraries. Their survival often depends on the ability to modify and adjust libraries according to final process design rules and parameters, as well as to shift a design to a different process by adjusting the library. This chapter describes the principal flow of a library conversion project based on real project data. Standard cell libraries, consisting of core cells and I/O cells, are discussed here. Memory and datapath compilers, which may be considered part of a library, are the subject of Chapter 10.

Because every library and every process is slightly different, this chapter does not focus on one specific library or process technology. The flow and information presented here is compiled from several projects. The problems, flow, and tasks described may not all be needed in one particular project.

11.1 Typical Standard Cell Structures

All standard cells in a given library have to be designed according to a library-specific template. A typical standard cell template is shown in Figure 11-1. Most libraries today place power busses at the top and bottom of the cell. The advantage is that power busses can be abutted, as shown in Figure 11-2, thereby doubling the power bus width. Other characteristics of the
template are the number of routing tracks between the power busses, single or double cell height, the way that nwell and pplus are drawn, the placement of substrate contacts, and the routing grid used.

![Possible positions for pickups](image)

*Figure 11-1. Typical standard cell template.*

The decision of how many routing tracks to use is often a philosophical one. Formerly, the number of routing tracks between power busses was determined by the most complex cell to be designed. This cell was usually a complex flip-flop. This approach works well when the place & route tool creates horizontal routing channels of variable height as shown in Figure 11-3. In a sea-of-gates approach, as shown in Figure 11-2, the P&R tool does not create extra routing channels for horizontal routing. In a sea-of-gates only the number of vertical routing tracks can be modified by inserting fill cells. This means that the number of horizontal routing tracks is defined by the number of tracks between the power busses. Therefore, some companies choose a higher number of tracks than are needed for designing the cells, because they believe that having more horizontal routing tracks will save overall area. Of course, the optimum number of tracks may change from one design to another, depending on a specific design’s routing requirements.

Another distinction is the way nwell and pplus are used inside the cells. Some templates use only rectangle-shaped nwell polygons, while others allow jogged nwell polygons, as shown in Figure 11-4. Handling jogged nwells is a bit more complex, because the check is more difficult, since all cells have to abut without violating design rules.
Figure 11-2. Standard cells placed without routing channels.

Figure 11-3. Routed standard cell design showing routing channels.
Most libraries require that sufficient substrate and well contacts are placed in each cell. Other flows create substrate and well contacts after routing is completed. Some architectures also strap substrate contacts under the power bus, while others share them between cells.

Another difference among libraries is the selection of routing pitch. If the design rule overhang via-metal is greater than 0, then it is possible to choose between a reduced pitch and a full routing pitch. As shown in Figure 11-5, the reduced pitch is smaller, but does not allow two vias to be placed next to each other on neighboring routing tracks, which restricts the cell design and the routing tool. This means that pickup ports cannot be placed on the farthest left and farthest right vertical routing tracks of a standard cell.

11.2 Special Standard Cell Library Conversion Requirements

Designing standard cells is more restrictive than, for example, designing full-custom cells. This is because a standard cell design must fulfill the requirements of EDA tools, and because a single library is designed for many different applications. Each library is thus always a compromise.
11.2.1 Considering all possible configurations

The conversion process must take into account the fact that each cell must fit in all possible configurations, i.e., that each cell can be combined with every other cell. This can be enforced by implementing worst-case rules for the layout design. However, to get a more optimum library in terms of layout density, more flexible rules are required which may increase the conversion and verification effort.

An example of such a rule is the abutting of nwell structures between neighboring standard cells. The design rules for minimum spacing and width of the nwell layer are usually large compared to the design rules for metal and poly structures. To prevent incorrect spacing and width situations of the nwell layer inside standard cell layouts, most libraries use a rectangular
nwell shape with the same height in each cell. Figure 11-4 shows three library cells with rectangular nwell structures. The area of the cells can potentially be reduced by allowing jogs inside the nwell layer, as shown in Figure 11-4b. The result is that the height of the cells can be reduced. However, implementing and verifying a solution using jogs requires more effort.

11.2.2 Special device sizing of output drivers

The transistors that are most important for the performance of a library, and for the performance and density of the circuits designed with that library, are the output drivers of the gate. If they are too small, the P&R tool has to insert additional drivers to refresh the signal for long wires. If the driver is too big, the cell becomes larger, and so do the final designs. A more ideal solution is a set of cells for each basic function. Each cell of the set uses a different transistor size to provide a different drive strength. Most libraries contain three to five cells of the most frequently used basic functions.

Some P&R systems work even better with a higher variety in drive strength. The latest concept is that of a liquid library. This means the cell with the required drive strength is generated on the fly, whenever the synthesis or P&R tool demands it. Cell architectures that support this concept place the drivers at the left or right side of the cell, where they can be easily expanded or contracted. The most flexible architectures are those in which driver transistors are horizontally arranged, as shown in Figure 11-6. Driver size can be easily modified by stretching the transistor.

Figure 11-6. Standard cell with horizontally arranged output drivers.
The library conversion process must support the ability to size transistors according to user-defined sizing functions. These functions reflect the electrical characteristics of the new process and the library's target applications.

### 11.2.3 Supporting a special grid for ports

Place & route tools require not only that library cells all have the same height, but also that the boundary and the pickup ports are on a special grid. This grid is usually a multiple of the basic technology grid; details have already been explained in Chapter 4, section 4.4.2.

### 11.3 Standard Cell Library Conversion Flow

This section describes the general conversion flow of a library of core standard cells. The conversion of I/O cells is discussed in section 11.4. When a physical layout is converted using ALM technology or similar techniques, the conversion of a core standard cell library of about 300 cells requires an effort of four to eight person-weeks. When the layout conversion process is done manually, it can take at least one day per cell, totaling 12 to 18 person-months. Aside from the huge difference in labor costs, automation's main advantage is the reduction in design time, and greater flexibility in choosing the target fab.

### 11.3.1 Enhancing the library

Library functions are not carved in stone: often, new functions are added. If an automatic conversion process is implemented, library design becomes an incremental process in which new cells are added over time to the common library template. Because the process is mostly automated, the additional costs of porting and maintaining new cells are minimal.

Some library design tools also offer the ability to synthesize the layout of additional cells according to a template. Although the area of these cells is often slightly larger than the area of cells which have been manually drawn, this is a good approach to begin with for creating the first layout of a cell.

### 11.3.2 Calculating device sizing and power bus width

Initial device sizing is based either on simulation or on evaluation of the process parameters. If the characteristics of the target process are similar to those of the current process, the devices are often sized linearly. This means the W/L ratio of each device is maintained in the new process. When the
gate length \((L)\) is reduced, the width \((W)\) of the device is reduced proportionately. If the new process is very different from the old process, the cells are simulated. The problem with simulating standard cells is that some simulation stimuli can only be estimated, specifically, the load attached to the outputs and inputs of the cells. These loads are estimated by performing statistical analysis of wire loads on typical designs. From this analysis, the size of the cells’ output drivers are derived.

Defining the right power bus width of a standard cell library is a bit like gazing into a crystal ball. The reason is very simple: the power consumption of a circuit designed with a library does not depend on the library itself. Instead, it depends on the frequency of the circuits; the number, capacity and distribution of high-frequency signals; the size of the circuit; and the global power grid. Since library designers often do not know these figures, they usually assume the worst-case scenario, which results in power busses often being wider than they need to be.

11.3.3 Adjusting the physical layout

The information in this section is compiled from projects that exclusively used ALM technology for layout adjustment. The LADEE Tool Suite from Rubicad Corporation was used for all projects. The main phases of the layout adjustment with this specific tool suite are:

- Extracting pickups, ports, power busses, and other cell properties.
- Converting or compacting the library.
- Selecting and enforcing the target cell height.

The extraction is usually done with mask and automatic editing operations. Often, the via cuts for the pickup have to be created because they are needed for the conversion, but are not part of the cell. The ports for power busses, and for nwell and pplus structures, have to be created so that the final position of these structures can be controlled and the final cells will match the template.

The setup of the conversion itself is similar to the general conversion flow described in Chapter 9. For a standard cell library conversion, different grid values for structures like pickup ports and frames must be considered. During the optimization phase, the number of via cuts for input and output signals are maximized to give the P&R tool a maximum number of choices.

The task that is most specific to standard cell libraries is enforcing a common cell height. This can be done in two ways. The first is by compacting or converting all cells in parallel and adjusting the cell height of
Aspects of Library Conversion Projects

all cells to the one with the maximum cell height. This solution is often not
chosen, because it is a worst-case solution.

The second way is to pre-select a common cell height and adjust all cells
to this height. Usually, the common cell height is either based on the desired
number of routing tracks or it is the minimum cell height which allows all
cells to be fitted. However, some cells may remain larger than the target cell
height. They are then separated and treated differently by modifying the
settings of the tool or by making some manual changes in the layout
arrangement.

The whole process of conversion takes about two to four person-weeks,
including setting up the tools and technology files, adjusting the cells, and
verification. The runtime is very short, normally between 30 seconds and
two minutes per cell, depending on the process, the library, and the
performance of the selected computer. In this way, an entire library can run
in three to four hours on a single 1-GHz CPU.

11.3.4 Verifying the physical layout

Verification is done with the usual DRC and LVS tools. The library-
specific part is considering the cells in the context of all other cells. This
means each cell has to be placed next to every other cell. To create such a
database for 300 cells, for example, would mean placing about 90,000 cells
in one layout. Although this is feasible, it is easier to make one special cell,
manually verify it, and place all 300 cells next to it, one by one. This would
only require about 900 placements, which is a very small database.

Besides DRC and LVS checks, it must be verified that all cells fulfill the
requirements of the P&R tool. These requirements are the grid of the frame
and the pickup vias, the position of pin names, cell names, and cell height.
These checks are done by scripts that run the check for the entire library.
Usually, these scripts are written by the design teams.

11.3.5 Characterizing library cells

The purpose of library characterization is to create timing models for the
synthesis and timing analysis tools. For that purpose, a SPICE netlist is
created from the schematic and simulated with different RC loads at the
input and output signals. Simulation results are used to create the timing
models, which are delay tables containing delay times for the different loads.

In UDSM processes, designers have to worry about crosstalk effects
caused by global wiring. In theory, the status of a flip-flop can be changed or
delayed by crosstalk caused by higher metal layers running over the cell. To
exclude this possibility, a worst-case situation is created by crossing the cell
with the maximum number of wires. In addition, stimuli are created for these wires. Parasitic devices are extracted from the thus-modified layout, and are simulated under worst-case conditions. If crosstalk effects are detected, the cell layout will be modified to prevent them.

The process of characterization is highly automated in most companies. It is done by generating SPICE runs with different stimuli and collecting the resulting data to produce the timing models and datasheets.

11.3.6 Updating views for EDA tools

For each cell, several views must be created. A view is a representation of a specific aspect of the cell for a specific tool. There are physical views, such as the P&R abstract, which is used as a placeholder for the cell during P&R and defines the cell’s dimensions as well as any routing restrictions. The routing view describes the pin positions and access direction of each pin of the cell. Some tools require an extraction view to extract parasitic wire capacitance from the final layout. Physical views are derived automatically from the physical layout.

Aside from physical views, logic views are also needed for synthesis, simulation, timing and power analysis, and verification. Examples are the SPICE netlist of the cell, a Verilog or RTL representation, and a timing representation in .LIB or ALF format. Most of these views are small text files containing the requested information in a specific format. Many tools import standard formats and use them to generate their own internal databases automatically.

The exact amount and types of views needed depend on the specific design flow and tool set, and is sometimes application-specific. However, most of the tasks can be automated using scripting techniques.

11.4 Converting I/O Cells

Every library must contain input and output cells to connect the chip to the outside world. These I/O cells are highly technology specific. They are often provided by the foundry, because they contain technology-specific electrostatic discharge (ESD) protection structures.

11.4.1 I/O cell structures

Each I/O cell contains a logic part and the input or output latch. The output latches are big driver structures to drive the external signal. They are designed using ESD structures and are connected to the bonding pad. The supply voltage level for these drivers can be different than for the internal
structures. The logic interface to the internal circuit is located on the G side of the cell (see Figure 11-7).

11.4.2 Converting ESD and bonding pad structures

The ESD structure of the pad cell is technology dependent and in addition, has analog characteristics. The target specification of the I/Os depends on the application for which the final device will be used. Therefore, the ESD parts of the I/Os are designed specifically for each library and are usually handcrafted. Fortunately, there are only a few ESD
structures per library, usually one or two, which are combined with the different logic interfaces.

11.4.3 Converting the logic parts of I/O cells

The logic parts of I/O cells can be converted in a manner similar to that of the core cells. However, cell height and cell architecture are different. They, too, have to be compatible with the routing grid and the P&R tool’s requirements. Therefore, the most efficient way to convert I/O cells is to convert the logic and ESD parts separately, and then combine them at the end of the conversion process.

11.5 Special-Purpose Libraries

Most libraries are designed to fit all possible applications. However, in SOC designs many different components are integrated on one device. Different components may demand different libraries for achieving more optimum designs. This means that the design of a single chip may require several different libraries for different voltage levels, circuit frequencies, and sizes, as well as for different power consumption levels.

The differences among these libraries are different transistor and driver sizes, cell heights, and sizes of power rails. These modifications can be implemented and supported by ALM technology, using an automatic conversion approach, resulting in a collection of libraries for each technology. When these libraries are combined into a single SOC design, the result is reduced power consumption and circuit size, and increased circuit speed.

11.6 Gate Array Libraries

Gate array designs are done with partly pre-processed wafers. Transistor gates are already produced in a fixed array on the wafer, which is then customized using only the metal layers. Gate array cells can be ported to different processes with ALM technology in a manner similar to standard cells. Since they contain only metal and via structures, diffusion, poly and transistor structures must be added for conversion purposes.
Appendix 1

**Key Algorithms Used for Automatic Layout Modification Technology**

One of the reasons that physical design reuse is not a widely accepted practice is because many people assume that a lot of manual layout work is required to implement such an approach. But this is not the case. To implement physical design reuse, ALM is needed. Physical design reuse is one of the primary applications for ALM, but ALM has many other applications in existing design flows and offers additional solutions to a large number of issues encountered in UDSM design.

The most common approaches to implementing ALM are logical mask processing and layout compaction tools. Logical mask processing capabilities are found in layout verification tools and mask tape-out software. Layout compaction is the process of increasing the density of a mask layout to reduce the overall silicon area needed to produce an IC. Reducing overall area is important because it reduces the production costs of a circuit in two ways. It increases the overall number of circuits per wafer, and it increases the number of good circuits, or yield, per wafer.

This chapter introduces different methods of automatic layout modification, and the different algorithms used for them. After explaining mask operations, a special focus is given to compaction algorithms.
A1.1 Mask Operations

Mask operations are functions performed on the entire mask. The term “mask” is used in different ways. One meaning is the physical object called an optical mask, which is used to create an optical image on the silicon wafer. The other meaning is equivalent to the term “layer”. In that sense, the mask is a set of polygons or a layer of polygons, which is used to derive the optical physical mask.

Mask operations can thus be described as operations on sets of layers. They are applied in DRC and LVS tools to identify design-rule errors and compare the layout with the schematic. Their primary tasks are to extract specific layout situations and identify electrical elements such as transistors, resistors and wires inside the layout. A set of functions includes logical functions, selection functions, sizing and layer operations, and check functions.

The nature of mask operations does not allow their application to detailed and very specific layout manipulations that require the movement of specific polygons or edges. This is because mask operations always work on an entire set of polygons. Mask operations do not consider design rules between masks, nor can they create additional space between polygons or edges. Algorithms associated with layout compaction are much more suitable for specific and detailed layout manipulations that operate on single polygons and edges, or when more space between structures must be created.

![Hierarchical mask operations](image)

*Figure A1-1. Hierarchical mask operations.*

Mask operations are usually defined for flat layout data. This is a limitation for big databases, which have to be handled hierarchically. The problem with hierarchical mask operations is illustrated in Figure A1-1. Not only must the polygons inside the cell be considered for the operation, but also the polygons drawn on a higher or lower hierarchy level. This means the layout has to be virtually flat. The more complex problem is to decide in
which hierarchy level the result has to be stored, so that the total amount of
data is minimized. Most tools usually handle mask operations as flat, and put
the resulting polygon on the top level.

A1.1.1 Logical mask operations

Logical mask operations are Boolean operations defined on mask layers.
A typical set of operations contains AND NOT, AND, OR and EXOR.
Figure A1-2 shows examples of the different Boolean operations.

![Logical mask operations](image)

Figure A1-2. Logical mask operations (Boolean operations).

Logical mask operations are used for the extraction tasks of identifying
electrical elements such as transistors, resistors, and wires inside the layout,
as well as for generating additional masks which can be derived from
existing masks.

The primary applications for mask operations are device extraction and
generation of additional or intermediate layers. A simple example of device
extraction is the identification of transistor gates. Transistor gates are formed
by the two layers of poly and diffusion inside the layout. To identify the
gates, a simple mask operation “poly and diffusion = gates,” as shown in
Figure A1-3, can be used. Complex processes with different devices – such
as N, P, high- and low-voltage devices, and sometimes bipolar devices – require longer sequences of mask operations to identify and distinguish all of the different devices.

Some sets also contain a NOT operation. The definition of the NOT operation requires a boundary layer covering the entire layout structure. The boundary layer defines the maximum extension of the layout. So NOT A means boundary AND NOT A. Without the boundary layer, the result of NOT A becomes infinitely large. Figure A1-4 shows the NOT operation.

**Figure A1-3. Simple mask operations.**

A1.1.2 Selection operations

Complex design rules require more than Boolean mask operations for checking design-rule correctness and extracting devices. Selection operations are needed to execute complex extraction tasks. A set of typical selection operations contains TOUCH, CROSS, INSIDE, OUTSIDE, ORTHOGONAL, NON ORTHOGONAL, RECTANGLE, NON RECTANGLE. These operations are best explained by giving some sample operations, such as those shown in Figure A1-5.

**Figure A1-4. NOT operation.**

A1.1.2 Selection operations
A1.1.3 Sizing operations

One of the most frequently used functions during mask processing is the sizing or biasing operation. The sizing function increases or reduces a polygon at the boundary. Examples are given in Figure A1-6.

This function is mainly used for biasing layers, removing notches, and checking distance violations.

As easy as the operation looks, it is not easy to implement and, in many cases, the function cannot be reversed. This means the sequence:

SIZE LayerA –10 micron; SIZE LayerA 10 micron is not equal to
SIZE LayerA 10 micron; SIZE LayerA –10 micron

as shown in Figure A1-7.

The sizing function is even defined in different ways in non-orthogonal structures. The problem here is that the polygon coordinates have to be snapped to the grid. So either the coordinates can be snapped to the grid, resulting in an inaccurate oversize value, or an accurate oversize value will be obtained but the coordinates will be off grid. Therefore, some systems
implement different versions of sizing functions to fulfill the different requirements.

![Figure A1-6. Typical Sizing Operations.](image)

A1.1.4 Layer operations

Layer operations include mainly the DELETE, COPY, MOVE, and MERGE functions. Aside from the MERGE function, all of the others should be obvious without further explanation. The MERGE function merges all connected polygons into one polygon, as shown in Figure A1-8. The only question is how the function deals with polygons that have Only one common point versus how it deals with self-intersecting polygons. Polygons with one common point are usually not merged; instead, they stay separate (see Figure A1-9). For polygons with self-intersections, there are two possible solutions: allowing self-intersection, or having an outer polygon and a set of inner polygons representing holes (see Figure A1-10). Which implementation is used depends on the application.

![Figure A1-7. Sizing operations cannot be reversed.](image)
Key Algorithms Used for ALM Technology

**Figure A1-8.** MERGE operation function.

**Figure A1-9.** Polygons with one common point do not merge.

**Figure A1-10.** MERGE function using polygons or areas.
A1.1.5 Check functions

Check functions are used to identify design-rule violations or to identify specific layout situations by measuring properties of polygons or between polygons. In the context of ALM, they are used as special selection functions to create sets of polygons. The main checks that can be made are:

- Distance between polygons of one or two different layers
- Width of polygons
- Overhang of one polygon over another polygon of a different layer
- Overlap between two polygons of different layers
- Area of a polygon

Examples of the different checks are given in Figure A1-11. Usually, the checks are labeling or flagging situations which are smaller than a minimum value.

The errors are marked with error polygons or error flags. The error flags can be used to select polygons and perform further operations on them.

![Figure A1-11. Sample check functions.](image)

A1.1.6 Implementation options for mask operations

Most mask operations are implemented inside verification systems for DRC and LVS checks. Most of these verification systems operate in batch mode. Some implementations are done in connection with polygon editors and compaction systems.

UDSM technologies require long extraction and checking routines composed of several hundred lines of code. These programs have to be verified for correctness, and often debugged when errors occur, because the correct implementation of DRC and LVS is critical for achieving a working chip on the wafer. For easier development and debugging of these routines
an interactive development environment is highly desirable. An interactive operation mode can provide a step-by-step mode that lets the user control each intermediate result immediately, without modifying the code as is required in batch mode.

**A1.2 Types of Compaction Algorithms**

Compaction algorithms have been around for more than 20 years. Most of them are related to graph algorithms. The algorithms can be classified according to key characteristics. The main elements of distinction are symbolic- and polygon-based data structures, one-dimensional or two-dimensional compaction, orthogonal and non-orthogonal data, and flat or hierarchical compaction.

**A1.2.1 Symbolic compaction – polygon compaction**

Two of the initial motivations for developing compaction software were to speed up the layout design process and to reduce its complexity. Therefore, many compactors provided a symbolic layout interface and the compaction algorithm worked on this higher level of data abstraction. For each schematic element, the symbolic layout interface provides a predefined layout cell. In this way, the layout representation is directly linked to the schematic. The advantages of this approach are faster layout input, fewer data elements and shorter runtime. However, the degree of compaction was limited by the simple representation of the elements. With the increasing complexity of design rules, more distinctions have to be made in the symbolic representation, making it eventually as complex as the layout presentation itself. Also, it is difficult to describe arbitrarily-shaped devices such as capacitors, meander-shaped transistors, and analog devices. Figure A1-12 shows an example of symbolic representation and mask representation.

The symbolic approach is limited by the fact that the layout requires a specific format, which is usually tool specific. Many layouts were only available as a mask layout, without a symbolic representation. Therefore, programs were developed to transfer a mask layout into a symbolic layout. However, these programs also had limitations. They could not handle non-orthogonal structures or meander-shaped transistors. Polygon compactors were developed to remove these restrictions.

Polygon compactors do the compaction directly on the polygon data. They can move polygons, or they can reshape them by moving polygon edges. In this way it is possible to compact mask data directly without first converting it into a symbolic representation, which is not an easy task when
considering 45-degree wires and meander-shaped transistors. In most cases, polygon compaction also produces a denser layout. Design rules can be implemented more precisely, because the level of abstraction is lower than in symbolic compaction. Examples of such design rules are butted tap or well contacts, gate-specific rules, and asymmetrical contact and via rules. In general, polygon and edge-based compaction can give the user a higher level of control than can symbolic compaction. Most advanced compaction tools in the market are polygon or edge-based tools. Both approaches can be implemented with similar compaction algorithms.

Figure A1-12. Symbolic representation versus mask representation.

A1.2.2 One-dimensional – two-dimensional

Compaction is defined as reducing or minimizing the layout area of a circuit implementation without violating the circuit’s functionality or specification and without creating design-rule violations. This means that it is a two-dimensional problem, and that the best results should therefore be possible when elements can be moved in both directions. However, most compaction algorithms are one-dimensional algorithms, because of the complexity in compute time required to achieve optimum two-dimensional compaction. A one-dimensional algorithm moves the element first along one axis and then, in a second phase, along the other axis.
However, two-dimensional algorithms do not necessarily produce better results, because of the look-ahead which must be performed for an optimum decision. This is demonstrated in the simple example shown in Figure A1-13. This example consists of simple squares with space between them. The task is to fill the spaces by moving the neighboring square from the space’s left or right into the empty space, with the ultimate goal of attaining the smallest overall area. To achieve this goal, one must perform a look-ahead to the upper right corner when making a move. This look-ahead is a non-linear operation and is therefore too compute-intensive for large designs.

However, as shown in Figure A1-14, the one-dimensional approach does not provide optimum results, either.

A solution that falls in between one- and two-dimensional algorithms is the 1.5-dimensional approach. The 1.5-dimensional algorithm moves elements in one dimension, but considers both dimensions for the decision. This approach provides reasonable runtime and good compaction results.

![Figure A1-13 Two-dimensional compaction.](image-url)
Actual implementation data show that two-dimensional compaction using linear programming and similar approaches can only be applied to very small problems, because of the required runtime. All algorithms that work on larger databases are one-dimensional.

\[ \text{Figure A1.14. One-dimensional compaction.} \]

\[ \text{A1.2.3 Orthogonal – non-orthogonal structures} \]

Compaction algorithms can also be classified according to their ability to handle non-orthogonal structures such as 45-degree wires and transistors. Most algorithms work efficiently only on orthogonal structures, while some can maintain 45-degree structures, and others can convert orthogonal structures into non-orthogonal ones and vice versa.

In the last few years, an increasing number of systems have been limited to an orthogonal layout structure, because it was thought that the benefits of 45-degree structures could not justify the complexity of the algorithms. However, 45-degree structures provide the advantages of smaller layouts, shorter wires, and better timing. Some physical layout structures, such as meander-shaped transistor structures, even require 45-degree structures.

\[ \text{A1.2.4 Flat compaction – hierarchical compaction} \]

The major challenge for compaction technology is the complexity of deep sub-micron chip designs. Designs now easily exceed the 10-million transistor level. This complexity can only be handled by hierarchical layout design, because of the size of the databases. Hierarchical compaction is therefore a necessary capability for handling large multimillion-gate designs.

Early approaches to hierarchical compaction were based on abutting, in which the layout had to tile into cells that abut each other. The compaction
tool processed each cell individually, which led to neighboring cells with different sizes. An abutting step then had to be executed to stretch the cells back to the same height. This type of compaction was executed on a two-level hierarchy. Multi-level hierarchy with overlapping cells requires far more complex approaches, including simultaneous and parallel compaction that considers all levels of hierarchy at the same time. These approaches are discussed at the end of this chapter in section A1.4. Most of the systems implemented today work with a flat layout or with a two- or three-level hierarchy.

A1.3 Compaction Algorithms in ALM

There are several approaches that have been used to apply compaction technology to automatic layout manipulation. The virtual grid and constraint graph approaches are well described in the available literature. There is not much literature available for the scan line compaction approach, because it is based on a proprietary algorithm implemented in commercial systems. However, the algorithm's properties are very interesting for their application to automatic layout manipulation.

A1.3.1 Shear line or virtual grid compaction

The shear line or virtual grid compaction approach was one of the earliest compaction algorithms. It is a raster-based algorithm. The principle idea is based on the observation that empty stripes in the layout occur which are covered only by horizontal connections. These empty stripes can be eliminated from the layout without changing its functionality. A refinement of the approach is that these stripes do not need to cover the entire width of the layout. Instead, it is enough if they cover different parts of it and can be connected with so-called shear lines.

The algorithm can be easy implemented by using a grid structure. All elements of the layout are placed on a raster, as shown in Figure A1-15. Elements which cannot be compressed are labeled “0” and those which can be compressed are labeled “1”. The strategy then becomes finding stripes of “1”s in the layout which can be eliminated.

The approach is one-dimensional, and therefore it has to be executed in x and y directions to achieve a complete compaction. Also, it is clear that multiple steps in the same direction will not improve the results. The quality of the results depends on the granularity of the grid. The finer the grid, the better the degree of compaction, but also the longer the runtime. If the grid is split in half, the number of grid points will quadruple. This will limit its application for large databases.
The algorithm can be applied to symbolic or polygon structures. The advantage of this algorithm is that it is very easy to understand and can be efficiently implemented.

Figure A1-15. Shear line, or virtual grid, compaction.
A1.3.2 Constraint graph compaction

One of the most frequently implemented approaches to compaction is constraint graph compaction. Constraint graph algorithms have been used and described since the late 1970s. This approach transforms the compaction problem into a graph problem. The layout symbols or objects are the nodes of the graph and the edges of the graph represent the constraints between the objects. The objects can be symbolic, or geometric such as polygons or edges. The time-consuming step of this approach is constructing the graph. The graph is then solved by finding its longest path, which determines the size of the layout in the direction of the graph. Therefore, the algorithm is called the longest-path algorithm.

Different methods can be used for constructing the graph. One of these is the shadow propagation method. This method consists of shining a light from behind the object under consideration and identifying all of the objects covered by the shadow of the object. Figure A1-16 shows a sample layout and the constraints between object A and its shadow objects.

![Figure A1-16. Constraints between object A and its shadow objects in sample layout.](image)

The constraints can be efficiently identified by using scan line algorithms. In general, the constraints in the graph represent the design rules for the layout. For a geometrical layout, these are the spacing, width, overhang, and overlap rules between the polygons in the mask layout. Figure A1-17 shows an example of a mask layout and its constraints graph. The constraints can be expressed in positive numbers, which are the design rule values between the layout edges.
A1.3.3. Iterative edge compaction

The scan line compaction algorithm is a one-dimensional polygon or edge-based compaction algorithm. It allows the compaction of raw polygon data without converting it into a symbolic representation. The smallest data unit which can be modified is an edge. The complexity of the algorithm is $O(n \log(n))$, which is faster than the $O(n^2)$ of an ideal longest path compaction algorithm. The expression $O(n)$ describes the complexity in relation to the input size. $O(n)$ means that the effort grows linear with the input size. $O(n^2)$ means that the effort grows with the square of the input size. So if the input doubles, the effort increases four-fold.
Iterative edge compaction was originally developed for layout migration to another process technology, but it can also be used for general compaction and layout optimization. The basic idea is very simple, and can best be illustrated by an example like the one shown in Figure A1-18.

The algorithm starts by placing the lowest left edge on the new position, and then removes the edge from the scan line. It then takes the next edge from the scan line, calculates the constraints to the already placed edges, and positions the new edge. That edge is then removed from the scan line and the next edge is chosen. This process stops when all edges are placed. In pseudo-code, the algorithm looks like the following:

```pseudo-code
While (edges in scan line) {
    Select next edge from scan line;
    Search constraints with already placed edges;
    Calculate new position of edge;
    Reconstruct polygon context;
    Remove edge from scan line;
}
```

To achieve a design-rule correct layout, the algorithm has to run in x and y directions. For orthogonal layouts, most of the tasks described above are
quite simple. The edge selection process goes from bottom to top and from left to right. Reconstructing the local polygon context and removing the edge from the scan line are very simple. The most complex task is finding the relevant constraints to calculate the new position of the edge. The simplest method would be to consider all edges already placed in the new layout. For large layouts, this would be too expensive in terms of runtime and memory resources. What is really needed is the upper fence of each layer. The upper fence is formed of the edges with the largest y-coordinates of the layout already processed. In this way, the number of edges to consider for the constraint search can be limited to a very small number.

The iterative edge compaction algorithm can be easily applied to non-orthogonal structures. In this case, the selection process is not as easy. The order of compaction is no longer from left to right and from bottom to top, as shown by the example in Figure A1-19. The extreme case is the complex cycle shown in Figure A1-20. In this case, no free edges can be found to select for compaction: all edges are constrained by other edges. The solution is simple: the edges a, b, c, and d have to be split up into two or more edges.

The real strength of the algorithm lies in its simplicity, as well as in having detailed control over its effects on the layout. Examples of this control are:

- The algorithm is an iterative process of moving edges. This enables an interactive observation not possible with other approaches. It makes interactive compaction possible.
- With this iterative approach, the selection of a subset of edges can be easily controlled. Selection criteria can be the orientation of the edge, the length of the edge, the layer, the position of the edge, and properties assigned to edges and polygons.
- The algorithm allows effective jogging of edges and conversion of edges from orthogonal to non-orthogonal orientation.
- The relative movement of the edges can be effectively controlled so that long wires can be prevented. Indeed, the algorithm can be controlled so that wires can be actively shortened.
- Special constraints can be used to implement wire-specific spacing and width to reduce metal migration and improve signal timing.

The control capabilities listed above make this algorithm suitable for a wide range of applications besides layout compaction and technology conversion. These applications include automatic design-rule error correction, yield optimization, timing correction and optimization, and crosstalk reduction.
A1.4 Compaction Algorithms for Full Hierarchical Designs

UDSM designs are represented by hierarchical layout data, because a flat representation would require too much data. Therefore, the compaction of UDSM designs requires a hierarchical approach. This is necessary because a flat compaction would need too much runtime and the resulting layout cannot be verified easily in the existing design flow. Hierarchical compaction is defined as the compaction of a layout that maintains the same hierarchical structure as the input layout: each device stays in the same cell on the same level of hierarchy. This excludes the possibility that cells will be
flattened, duplicated, merged, or split up, since these modifications would change the original hierarchical structure of the layout. The original layout hierarchy may contain overlapping cells and any number of hierarchy levels.

The implications of this definition are that each cell must fit in every context or placement where it is used, it can only exist once in the total layout, and no duplication of a cell is allowed. However, it can be mathematically proven that there are hierarchical compaction problems for which no solution exists without modifying or exploding the hierarchy. This means the layout is constrained by the new design rules so that some cell instances have to be copied to a different name and changed differently or exploded to get a design-rule-correct and LVS-correct layout.

In the example given in Figure A1-21, when the cell A is compacted the routing passing through A in the different placements must be considered. These routings are called overlays on a higher level of hierarchy. Creating and extracting the overlays for each cell are complex tasks. In order to perform these tasks, a layout analysis tool is required which extracts the overlays from all hierarchy levels and stores them in a special database. The references to the original location of the overlays must also be stored inside the database.

![Figure A1-21 Extracting overlays of a cell by using hierarchical analyses.](image)

After the layout is analyzed, the cells can be compacted while considering all of the overlays in the different hierarchy levels. However, the connections to neighboring cells and the cell abutments have not yet been considered. To do this, a parallel compaction approach is required that
synchronizes the port positions on the fly. In Figure A1-22, parallel compaction is sequentialized for three cells. The parallel compaction approach is more effective than the sequential approach because it requires only one pass, and it gives better results because it provides more flexibility and therefore creates smaller cells.

Figure A1-22. Parallel compaction sequentialized for three cells.
The final step is the reconstruction of the entire layout using the compacted cells and the information extracted in the layout analysis phase. This approach has been used successfully for memory compilers and datapaths.

A1.5 Limitations of Compaction

Compaction has some inherent limitations. The most obvious is that compaction cannot totally reshape the global outline of a block of layout, as shown in Figure A1-23. It is possible to change the aspect ratio of a block gradually, but not in an extreme way. In addition, reshaping is not desired in most applications because it may affect the circuit's timing and functionality, since reshaping the layout modifies the length of the wires relative to each other.

![Figure A1-23. Simple example of limitations of compaction.](image)

A function many designers like to have in combination with layout compaction is to re-route a layout circuit and change the overall layout architecture of the layout circuit (see Figure A1-24). This cannot be achieved by compaction. It is definitely impossible to compact a layout with four metal layers into a layout with only three metal layers. Neither is it possible to change the entire floorplan of a layout using compaction. However, it is possible to use additional metal layers when available. So it may be possible to convert a layout using three metal layers into one using four metal layers. This may also improve the degree of compaction.

Many compaction systems cannot control the degree of compaction on a fine scale. The effect is that layout blocks containing intentionally “loose” areas may be compacted too much, which in turn causes timing and signal integrity problems. Some systems can prevent this by automatically controlling the degree of compaction and by inserting signal-specific constraints.
A1.6  Layout Correction Algorithms

One application that is very closely related to compaction is layout correction. Layout correction means the correction of DRC and LVS errors, as well as the correction of timing and power by doing specific transistor sizing and wire sizing. The main difference between correction and compaction is that correction is used to correct the errors by changing the layout as little as possible.
A1.6.1 Mask operations for design-rule correction

The first approach to automatic correction was, and is in most cases, the use of mask operations provided by verification tools. This approach is suitable as long as objects do not need to be moved. Moving objects such as contacts or wires cannot be achieved by logical mask operations and sizing operations, because all data on the layer is affected in the same way. This approach to automatic correction was mainly part of mask layer adjustments in combination with a linear shrink. In a situation such as the endcap violation shown in Figure A1-25, a correction using mask operations is not possible. A more intelligent correction approach is needed.

![Diagram showing endcap violation and corrections](image)

*Figure A1-25. Mask operations are not sufficient for automatic layout correction.*

The hierarchical design-rule correction cannot be handled with mask operations, because only the current cell can be seen. An example is shown in Figure A1-26. Hierarchical correction requires a much more sophisticated approach when flattening and renaming of cells must be prevented. Therefore, much of the design-rule correction is done manually.
A1.6.2 Grid snapping error correction after a linear shrink

A linear shrink on the mask layout level requires a grid snap of the layout structures. This grid snap can cause DRC and LVS errors when done automatically without considering each specific layout situation. The reason for these differences in snapping is the layout hierarchy and the snapping of the cell reference coordinate. When a cell reference is snapped to grid, then the contents of the cell are moved. This can cause two cells, which abutted before the shrink, to not abut after it, thus creating an open signal.

To solve this problem, either a more sophisticated grid snapping for hierarchical layouts must be used, or the layout must be flattened and the opens corrected by adding polygon pieces on the top level. The last solution is feasible for medium-sized layouts. For large databases, flattening the layout database is out of the question because they cannot be processed in a reasonable amount of time.

A1.6.3 Correcting hierarchical layouts

The main challenge in hierarchical layout correction is demonstrated in Figure A1-27. This shows a cell A placed several times in the layout. Layout structures drawn on top of cell A in a different hierarchy level or from overlapping cells are different in each instance. The task is to correct the
design-rule errors by fixing cell A in such a way that it can stay the same cell in all instances.

For achieving this, a new view of cell A is required that contains all of the environments of cell A, so they can all be considered at the same time. Figure A1-28 shows such a view.

Aside from fixing design-rule errors, the hierarchical layout correction approach can also be used to solve other problems. Applications include optimizing power consumption by changing transistor sizes and wire sizes, improving optical proximity correction (OPC) and phase shift mask (PSM) creation, or solving crosstalk problems. The overall benefits of this method are that polygons and edges can be moved individually without violating design rules or flattening the hierarchy.
Figure A1-28. View of cell A with "overlays" from all instances.
Bibliography


13. C. Mead, L. Conway; Introduction to VLSI Systems, Addison Wesley 1980

14. K. Mehlhorn; Data Structures and Algorithms 2: Graph Algorithms and NP Completeness, Springer 1984


18. R. Brueck; Entwurfswerkzeuge fuer VLSI Layout, Hanser Verlag 1993


### Glossary

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A/D</td>
<td>Analog-to-Digital converter</td>
</tr>
<tr>
<td>ALM</td>
<td>Automatic Layout Modification of IC mask layout</td>
</tr>
<tr>
<td>AP &amp; R</td>
<td>Automatic Place &amp; Route</td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>ASIC</td>
<td>Application-Specific Integrated Circuit</td>
</tr>
<tr>
<td>BiCMOS</td>
<td>Bipolar Complementary Metal-Oxide Semiconductor</td>
</tr>
<tr>
<td>CAD</td>
<td>Computer Aided Design</td>
</tr>
<tr>
<td>CMOS</td>
<td>Complementary Metal-Oxide Semiconductor</td>
</tr>
<tr>
<td>CPU</td>
<td>Central Processing Unit</td>
</tr>
<tr>
<td>D/A</td>
<td>Digital-to Analog converter</td>
</tr>
<tr>
<td>Design rules</td>
<td>Rules that constrain IC mask layout topology to assure fabrication process compatibility and yield.</td>
</tr>
</tbody>
</table>
Glossary

 DSP  Digital Signal Processor. A high-speed, general-purpose arithmetic unit used for performing complex mathematical operations.

 ECO  Engineering Change Order

 EDA  Electronic Design Automation

 EDIF  Electronic Data Interchange Format

 EEPROM  Electronically Erasable Programmable Read-Only Memory. A non-volatile memory that retains data after power is turned off, and can be erased and reprogrammed in place.

 EPROM  Erasable Programmable Read-Only Memory

 Flash memory  A non-volatile memory that retains data after power is turned off, and can be erased and reprogrammed in place. Denser and less expensive than EEPROM.

 Gate  Basic circuit that produces an output when certain input conditions are satisfied.

 GDSII  Graphical Design System II. Industry-standard data format for exchanging IC physical design data.

 HDL  Hardware Description Language

 IDM  Integrated Device Manufacturers

 IP  Intellectual Property

 NRE  Non-Recurring Engineering (costs)

 OPC  Optimal Proximity Correction

 PDR  Physical Design Reuse of IC mask layout

 PLL  Phase-Locked Loop

 PROM  Programmable Read-Only Memory. Programmed once by the buyer.

 PSM  Phase Shift Mask

 RAM  Random Access Memory

 ROM  Read-Only Memory. Programmed once by chip manufacturer

 RTL  Register-Transfer Level

 SCI  Serial Communication Interface
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOC</td>
<td>System-On-a-Chip</td>
</tr>
<tr>
<td>SPI</td>
<td>Synchronous Peripheral Interface</td>
</tr>
<tr>
<td>UDSM</td>
<td>Ultra Deep Sub-Micron technology with feature sizes of 0.18-micron or below.</td>
</tr>
<tr>
<td>VHDL</td>
<td>Very High-Level Description Language</td>
</tr>
</tbody>
</table>
Index

45-degree, 60
transistor, 105
design rules, 61

A
Acute angles, 103
Algorithms for compaction, 191-209
Alpha CPU, 133
Analog layout, 99
Analog structures
  identifying, 108
  separating, 107
Analysis tools, 119
  parasitic extraction, 120
  power, 125
  RC delay, 124
  timing analysis, 123
Antenna rules, 17
ASIC design flow, 89, 90
  integration, 77
  iterations, 78-80
  physical design reuse, 81
  typical, 78
Automatic layout conversion, 25
  design flow, 25
  parasitic device, 26
  transistor sizing, 26
  wire sizing, 26
Automatic layout modification
  libraries, 68
  characteristics, 51
  functionality, 51

B
Back-annotation, 78-79
Bit cell, 161
Bottleneck in layout, 16

C
Capacitance
  controlling, 39, 52
  coupling, 40
fringe, 85-86
lateral, 85-86
vertical, 85-86
Cell abutting, 108
Changing gate length, 27
Characterization of standard cells, 179
Chip analysis, 34
Chip-less, 2
Circuit errors, 46
Clock
frequency, 40
skew, 40
tree, 40
CMOS technology
strength and weakness, 23
Conductance, 30-32
Connectivity, 62
Contact
buried, 117
butted tap, 116
diamond structure, 104
maximizing, 62, 63
minimizing, 62
octagonal-shaped, 104
shared, 116
splitting, 62
pseudo, 152
Controlling
capacitance, 52
resistors, 52
Critical delay path, 38
Critical features, 45
Critical signals, 35-36, 56
extraction, 122
filtering, 41
separation, 57
shielding, 55
Crosstalk, 40, 125
dynamic effects, 42
reducing, 42
Current flow, 58

D
Data integrity, 73
Database
processing parallel, 72
Deadlock, 111
De-compaction, 21
Delay effects, 31
Delay time, 32
comparison, 37
Design automation gap, 10
Design challenge
signal integrity, 10
timing, 10
Design cycle, 11
Design flow
checking, 153
correcting, 153
hardware, 9
IC, 11
iterations, 20
software, 9
Design guidelines, 103
Design improvement, 87
Design iterations, 87
Design optimization, 43
power, 43, 127
timing, 43, 126
Design partitioning, 80
Design reuse
benefits, 1
hard IP, 13
hard macros, 2
linear shrink, 4
shrink, 94
Design size, 8, 31
Design style, 91
Design time
manual layout, 22
Design types
analog, 99
digital, 91
Design rules
coding, 74
correction, 18, 86
entry, 74
increasing, 45
recommended, 66, 73
relaxation, 66
Device sizing, 20, 43, 88
output driver, 176
signal-specific, 52
sizing instruction, 52
Distributed processing, 72
Drawing grid, 24
Drawing layout, 87
DRC, 18, 131
DSM
characteristics, 31-33

EDA
infrastructure, 2
Electrical characteristics
migration of, 51
Electrical compatibility, 94
Electrical parameters, 84
geometrical structures, 52
Electrical relationship, 84
Electro-migration, 45
Endcap, 151
End-of-line rule, 48, 149
Epitaxial wafers, 63
Error location, 132
ESD structure, 180
Extraction
1-dimensional, 35
2-dimensional, 35
3-dimensional, 35
capacitance, 20, 35
critical signals, 122
devices, 149
hierarchy, 146
methods, 120
netlist, 75, 149
resistors, 20, 35
wire width, 58

F
Fabless semiconductor companies, 96
Fan-out, 78
Feedback loop, 79
Floorplanning, 112
Fringe capacitance, 40
Full-custom design
digital, 94, 100
Full-custom blocks
reusing, 114
Full-custom layout, 16, 17

G
Gate array, 182
Geometrical relationship, 83
Geometrical requirements
of process technology, 82
Grid
centered, 69
size, 87

H
Hardware description language, 9
Hardware design, 9
HDL, see Hardware description language
Hierarchical analysis, 74
Hierarchical topology, 73
Hierarchical correction, 141
Hierarchical modification, 73, 108-112
array structures, 110
cell abutting, 108
deadlock, 111
matching hierarchy, 112
mirrored cells, 110
requirements, 108
rotated cells, 110
Hierarchy
device assignment, 105
memory, 164
optimization, 145
reconstructing layout, 155
High-frequency circuit, 92
High-threshold voltage mask, 140

I
I/O cell, 180
Image simulation, 46
Improving
connectivity, 62
IP, 2
integration, 97
soft, 7
IR drop, 58

J
Jog
inserting, 59
minimum length, 59
removing, 59

L
Lambda approach, 143
Latch-up, 63
Latch-up, input/output, 180
Lateral capacitance, 39
Layout
design time, 22
drawing, 87
full-custom, 16, 17, 100
mixed-signal, 100
partitioning, 112
reconstructing, 155
Layout adjustment, 88
Layout analysis, 74
Layout compaction
analog, 18
digital, 18
Layout conversion, 82
memory, 159
Layout correction
Alpha CPU, 143
error reduction, 157
Layout creation
automatic, 15, 16
Layout customization
different voltage applications, 26
gate length, 26
gate width, 26
libraries, 30
Layout design
analog structures, 117
bottleneck, 16
Layout generators, 16
Layout guidelines, 103
Layout migration, 22
Layout modification
algorithms, 183
Alpha CPU, 133, 142
design flow, 133
design-rule correction, 137
macro block, 91, 133
UDSM design rules, 137
Layout optimization, 43, 67
circuit speed, 28
power consumption, 88
yield, 88
Layout quality, 59
Layout tools
compaction, 130
editing, 129
modification, 130
open database, 130
place & route, 128
verification, 131
Leaf cell, 162
Level of abstraction, 81
Linear shrink, 4, 19, 23, 82
hierarchical, 143
Local interconnect, 152
Low power, 28
LVS, 131
M
Macro block, 91
Mask, adding, 30
Mask making
  costs, 46
Mask modification, 46
Mask operations, 19, 184-191
  Boolean, 75
Memory, 99
  bit cell, 161
  conversion requirements, 161, 165
  leaf cell, 162
  multiple shrink factor, 164
  shrink factor analysis, 168
Memory compiler, 16
  common library, 164
Mixed-signal, 99
Mobile applications, 28
Module generator, 16

N
Netlist information, 75
Noise, 125
  reduction, 40
Non-orthogonal layout, 60

O
Optical proximity correction, 46
  avoidance, 48
  rule-based, 46
OPC, see Optical proximity correction
Optimization tools, timing, 126
Orthogonal layout, 60

P
Parasitic device, 52
Partitioning, 112
  of layout, 82
Phase shift mask, 46, 49
  creation, 49
Physical compiler
  technology-independent, 102
  Physical design reuse, 81
  Physical implementation, 78
  Pick-ups, 69
    position, 69, 71
  Pitch
    full, 174
    reduced, 174
  Place & route, 15
    tools, 128
Platform design, 12
Portability, 96
  designing layout for, 114
  guidelines, 114
Power optimization, 43, 127
Power bus, 64, 70
  sizing, 70
  width, 177
Power consumption, 28, 43, 44, 67, 88
Power distribution, 43
Power grid, 64
Pre-defined circuits, 81
Process compatibility, 23
Process development time, 96
Process options, 30
Process parameter, unstable, 13
Productivity gap, 1
Pseudo-contacts, 152
PSM, 46

R
RC delay comparison, 37
Reconstructing layout, 155
Redundancy, 45
Relaxing, critical features, 45
Re-layout,
  full-custom, 22
  limitations, 22
Reliability factor, 44
Reliability problem, 45
Resistance, 85
Resistors, controlling, 52
Resolution enhancement, 49
Routability, 71
Routing pitch, 69

S
Scalable, 23
Sea-of-gates, 172
Self-intersecting polygons, 103
Shielding, 55
Shrink factor, 83, 95
  analysis, 168
  multiple, 163
Shrinkable design, 83
Signal integrity, 40, 55, 87, 125
Signal-specific
  layout manipulation, 55
  sizing, 54
Simulation, 34, 84
Slitting wires, 66
SOC, 7, 91
  complexity, 8, 12, 14
  design, 96
  customization, 14
Software design, 9
Standard cell
  height, 70
  pick-ups, 172
Standard cell block, 92
  reusing, 113
Standard cell design
  routing channels, 181
Standard cell library
  cell height, 100
  checking, 71
  conversion, 171
  conversion flow, 177
  conversion requirement, 174
  converting, 100
  customization, 68
  evaluating, 71
  grid, 100
  multiple grid, 69
  power bus, 101
Static timing analysis, 36
Structured full-custom
datapath, 98
EPROM, 98
FPGA, 98
RAM, 98
ROM, 98

T
Target process, 94
Technology-independent, 102
Text
  conventions, 112
  positioning, 104
Timing, optimizing, 43, 126
Timing analysis
  dynamic, 123
  static, 123
  tools, 123
Timing closure, 30, 95
  problems, 34
  wire delay, 31
Timing errors
  hold time, 38
  setup time, 38
Transistor, 45-degree, 105
Transistor sizing, 28, 44
  adding fingers, 38
  deleting fingers, 38
  prevented, 105

V
Verification tools
  DRC, 131
  error location, 132
  LVS, 131
  timing, 132
Vertex, 46, 49
VHDL, 78
Via
  maximizing, 62, 63
  splitting, 63

via
Via cuts, 52
Voltage drop effects, 42

W

Wire
capacitance, 54, 85
critical, 55
delay, 20, 31-33, 79
moving, 57
profile, 32

resistance, 85
signal-specific, 54
sizing, 28, 43-44, 55, 64, 70
slitting, 66
spacing, 54, 56
width, 70

Yield, 88
critical features, 45
improvement, 66
optimization, 44