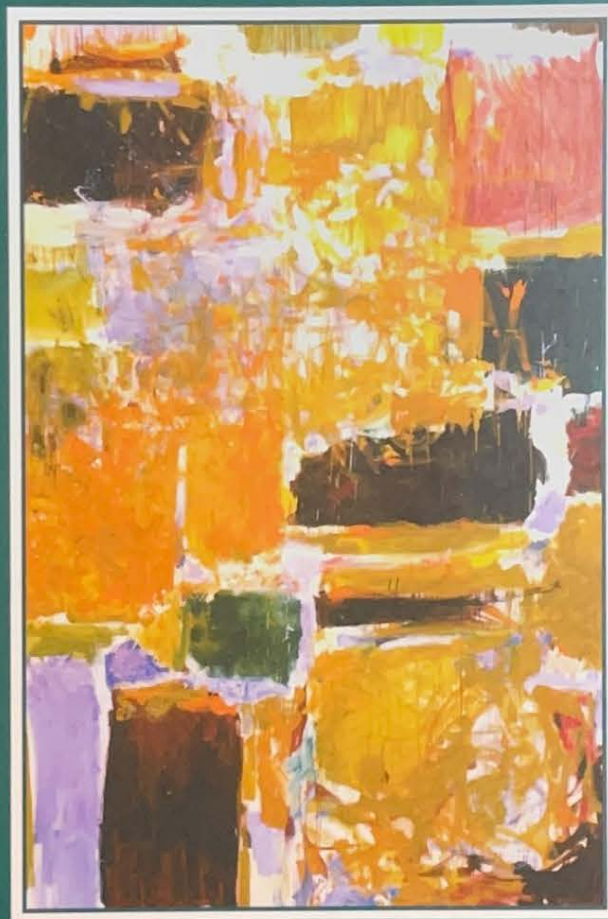


DIGITAL INTEGRATED CIRCUITS

A DESIGN PERSPECTIVE

SECOND EDITION



JAN M. RABAEY
ANANTHA CHANDRAKASAN
BORIVOJE NIKOLIĆ

PRENTICE HALL ELECTRONICS AND VLSI SERIES
CHARLES G. SODINI, SERIES EDITOR

DIGITAL INTEGRATED CIRCUITS

**A DESIGN PERSPECTIVE
SECOND EDITION**

**JAN M. RABAEY
ANANTHA CHANDRAKASAN
BORIVOJE NIKOLIĆ**

PRENTICE HALL ELECTRONICS AND VLSI SERIES
CHARLES G. SODINI, SERIES EDITOR



Pearson Education, Inc.
Upper Saddle River, New Jersey 07458

Library of Congress Cataloging-in-Publication Data on file.

Vice President and Editorial Director, ECS: *Marcia J. Horton*
Publisher: *Tom Robbins*
Editorial Assistant: *Eric Van Ostenbridge*
Vice President and Director of Production and Manufacturing, ESM: *David W. Riccardi*
Executive Managing Editor: *Vince O'Brien*
Managing Editor: *David A. George*
Production Editor: *Daniel Sandin*
Director of Creative Services: *Paul Belfanti*
Creative Director: *Carole Anson*
Art and Cover Director: *Jayne Conte*
Art Editor: *Greg Dulles*
Manufacturing Manager: *Trudy Piscioti*
Manufacturing Buyer: *Lisa McDowell*
Marketing Manager: *Holly Stark*

About the Cover: Detail of "Wet Orange," by Joan Mitchell (American, 1925–1992). Oil on canvas, 112 × 245 in. (284.5 × 622.3 cm). Carnegie Museum of Art, Pittsburgh, PA. Gift of Kaufmann's Department Store and the National Endowment for the Arts, 74.11. Photograph by Peter Harholdt, 1995.



© 2003, 1996 by Pearson Education, Inc.
Pearson Education, Inc.
Upper Saddle River, NJ 07458

The author and publisher of this book have used their best efforts in preparing this book. These efforts include the development, research, and testing of the theories and programs to determine their effectiveness. The author and publisher shall not be liable in any event for incidental and consequential damages in connection with, or arising out of, the furnishing, performance, or use of these programs.

All rights reserved. No part of this book may be reproduced, in any form or by any means, without permission in writing from the publisher.

Printed in the United States of America

10 9 8 7 6 5 4 3 2

ISBN 0-13-090996-3

Pearson Education Ltd., *London*
Pearson Education Australia Pty. Ltd., *Sydney*
Pearson Education Singapore, Pte. Ltd.
Pearson Education North Asia Ltd., *Hong Kong*
Pearson Education Canada Inc., *Toronto*
Pearson Educación de México, S.A. de C.V.
Pearson Education—Japan, *Tokyo*
Pearson Education Malaysia, Pte. Ltd.
Pearson Education Inc., *Upper Saddle River, New Jersey*

Contents

Preface	vii
Part 1 The Fabrics	1
Chapter 1 Introduction	3
1.1 A Historical Perspective	4
1.2 Issues in Digital Integrated Circuit Design	6
1.3 Quality Metrics of a Digital Design	15
1.3.1 Cost of an Integrated Circuit	16
1.3.2 Functionality and Robustness	18
1.3.3 Performance	27
1.3.4 Power and Energy Consumption	30
1.4 Summary	31
1.5 To Probe Further	31
Reference Books	32
References	33
Chapter 2 The Manufacturing Process	35
2.1 Introduction	36
2.2 Manufacturing CMOS Integrated Circuits	36
2.2.1 The Silicon Wafer	37
2.2.2 Photolithography	37
2.2.3 Some Recurring Process Steps	41
2.2.4 Simplified CMOS Process Flow	42
2.3 Design Rules—The Contract between Designer and Process Engineer	47
2.4 Packaging Integrated Circuits	51
2.4.1 Package Materials	52
2.4.2 Interconnect Levels	53
2.4.3 Thermal Considerations in Packaging	59
2.5 Perspective—Trends in Process Technology	61
2.5.1 Short-Term Developments	61
2.5.2 In the Longer Term	63
2.6 Summary	64

2.7	To Probe Further	64
	References	64
Design Methodology Insert A IC LAYOUT		67
A.1	To Probe Further	71
	References	71
Chapter 3 The Devices		73
3.1	Introduction	74
3.2	The Diode	74
3.2.1	A First Glance at the Diode—The Depletion Region	75
3.2.2	Static Behavior	77
3.2.3	Dynamic, or Transient, Behavior	80
3.2.4	The Actual Diode—Secondary Effects	84
3.2.5	The SPICE Diode Model	85
3.3	The MOS(FET) Transistor	87
3.3.1	A First Glance at the Device	87
3.3.2	The MOS Transistor under Static Conditions	88
3.3.3	The Actual MOS Transistor—Some Secondary Effects	114
3.3.4	SPICE Models for the MOS Transistor	117
3.4	A Word on Process Variations	120
3.5	Perspective—Technology Scaling	122
3.6	Summary	128
3.7	To Probe Further	129
	References	130
Design Methodology Insert B Circuit Simulation		131
	References	134
Chapter 4 The Wire		135
4.1	Introduction	136
4.2	A First Glance	136
4.3	Interconnect Parameters—Capacitance, Resistance, and Inductance	138
4.3.1	Capacitance	138
4.3.2	Resistance	144
4.3.3	Inductance	148
4.4	Electrical Wire Models	150
4.4.1	The Ideal Wire	151
4.4.2	The Lumped Model	151
4.4.3	The Lumped RC Model	152
4.4.4	The Distributed rc Line	156
4.4.5	The Transmission Line	159

4.5	SPICE Wire Models	170
4.5.1	Distributed rc Lines in SPICE	170
4.5.2	Transmission Line Models in SPICE	170
4.5.3	Perspective: A Look into the Future	171
4.6	Summary	174
4.7	To Probe Further	174
	References	174
Part 2	A Circuit Perspective	177
Chapter 5	The CMOS Inverter	179
5.1	Introduction	180
5.2	The Static CMOS Inverter—An Intuitive Perspective	180
5.3	Evaluating the Robustness of the CMOS Inverter: The Static Behavior	184
5.3.1	Switching Threshold	185
5.3.2	Noise Margins	188
5.3.3	Robustness Revisited	191
5.4	Performance of CMOS Inverter: The Dynamic Behavior	193
5.4.1	Computing the Capacitances	194
5.4.2	Propagation Delay: First-Order Analysis	199
5.4.3	Propagation Delay from a Design Perspective	203
5.5	Power, Energy, and Energy Delay	213
5.5.1	Dynamic Power Consumption	214
5.5.2	Static Consumption	223
5.5.3	Putting It All Together	225
5.5.4	Analyzing Power Consumption Using SPICE	227
5.6	Perspective: Technology Scaling and its Impact on the Inverter Metrics	229
5.7	Summary	232
5.8	To Probe Further	233
	References	233
Chapter 6	Designing Combinational Logic Gates in CMOS	235
6.1	Introduction	236
6.2	Static CMOS Design	236
6.2.1	Complementary CMOS	237
6.2.2	Ratioed Logic	263
6.2.3	Pass-Transistor Logic	269
6.3	Dynamic CMOS Design	284
6.3.1	Dynamic Logic: Basic Principles	284
6.3.2	Speed and Power Dissipation of Dynamic Logic	287

6.3.3	Signal Integrity Issues in Dynamic Design	290
6.3.4	Cascading Dynamic Gates	295
6.4	Perspectives	303
6.4.1	How to Choose a Logic Style?	303
6.4.2	Designing Logic for Reduced Supply Voltages	303
6.5	Summary	306
6.6	To Probe Further	307
	References	308
<i>Design Methodology Insert C How to Simulate Complex</i>		
	Logic Circuits	309
C.1	Representing Digital Data as a Continuous Entity	310
C.2	Representing Data as a Discrete Entity	310
C.3	Using Higher-Level Data Models	315
	References	317
<i>Design Methodology Insert D Layout Techniques for Complex Gates</i>		319
Chapter 7	Designing Sequential Logic Circuits	325
7.1	Introduction	326
7.1.1	Timing Metrics for Sequential Circuits	327
7.1.2	Classification of Memory Elements	328
7.2	Static Latches and Registers	330
7.2.1	The Bistability Principle	330
7.2.2	Multiplexer-Based Latches	332
7.2.3	Master-Slave Edge-Triggered Register	333
7.2.4	Low-Voltage Static Latches	339
7.2.5	Static SR Flip-Flops—Writing Data by Pure Force	341
7.3	Dynamic Latches and Registers	344
7.3.1	Dynamic Transmission-Gate Edge-triggered Registers	344
7.3.2	C ² MOS—A Clock-Skew Insensitive Approach	346
7.3.3	True Single-Phase Clocked Register (TSPCR)	350
7.4	Alternative Register Styles*	354
7.4.1	Pulse Registers	354
7.4.2	Sense-Amplifier-Based Registers	356
7.5	Pipelining: An Approach to Optimize Sequential Circuits	358
7.5.1	Latch- versus Register-Based Pipelines	360
7.5.2	NORA-CMOS—A Logic Style for Pipelined Structures	361
7.6	Nonbistable Sequential Circuits	364
7.6.1	The Schmitt Trigger	364
7.6.2	Monostable Sequential Circuits	367
7.6.3	Astable Circuits	368
7.7	Perspective: Choosing a Clocking Strategy	370
7.8	Summary	371

Contents	xix
7.9 To Probe Further	372
References	372
Part 3 A System Perspective	375
Chapter 8 Implementation Strategies for Digital ICS	377
8.1 Introduction	378
8.2 From Custom to Semicustom and Structured-Array Design Approaches	382
8.3 Custom Circuit Design	383
8.4 Cell-Based Design Methodology	384
8.4.1 Standard Cell	385
8.4.2 Compiled Cells	390
8.4.3 Macrocells, Megacells and Intellectual Property	392
8.4.4 Semicustom Design Flow	396
8.5 Array-Based Implementation Approaches	399
8.5.1 Prediffused (or Mask-Programmable) Arrays	399
8.5.2 Prewired Arrays	404
8.6 Perspective—The Implementation Platform of the Future	420
8.7 Summary	423
8.8 To Probe Further	423
References	424
Design Methodology Insert E Characterizing Logic and Sequential Cells	427
References	434
Design Methodology Insert F Design Synthesis	435
References	443
Chapter 9 Coping with Interconnect	445
9.1 Introduction	446
9.2 Capacitive Parasitics	446
9.2.1 Capacitance and Reliability—Cross Talk	446
9.2.2 Capacitance and Performance in CMOS	449
9.3 Resistive Parasitics	460
9.3.1 Resistance and Reliability—Ohmic Voltage Drop	460
9.3.2 Electromigration	462
9.3.3 Resistance and Performance—RC Delay	464
9.4 Inductive Parasitics*	469
9.4.1 Inductance and Reliability—Voltage Drop	469
9.4.2 Inductance and Performance—Transmission-line Effects	475
9.5 Advanced Interconnect Techniques	480

9.5.1	Reduced-Swing Circuits	480
9.5.2	Current-Mode Transmission Techniques	486
9.6	Perspective: Networks-on-a-Chip	487
9.7	Summary	488
9.8	To Probe Further	489
	References	489
Chapter 10	Timing Issues in Digital Circuits	491
10.1	Introduction	492
10.2	Timing Classification of Digital Systems	492
10.2.1	Synchronous Interconnect	492
10.2.2	Mesochronous interconnect	493
10.2.3	Plesiochronous Interconnect	493
10.2.4	Asynchronous Interconnect	494
10.3	Synchronous Design—An In-depth Perspective	495
10.3.1	Synchronous Timing Basics	495
10.3.2	Sources of Skew and Jitter	502
10.3.3	Clock-Distribution Techniques	508
10.3.4	Latch-Based Clocking*	516
10.4	Self-Timed Circuit Design*	519
10.4.1	Self-Timed Logic—An Asynchronous Technique	519
10.4.2	Completion-Signal Generation	522
10.4.3	Self-Timed Signaling	526
10.4.4	Practical Examples of Self-Timed Logic	531
10.5	Synchronizers and Arbiters*	534
10.5.1	Synchronizers—Concept and Implementation	534
10.5.2	Arbiters	538
10.6	Clock Synthesis and Synchronization Using a Phase-Locked Loop*	539
10.6.1	Basic Concept	540
10.6.2	Building Blocks of a PLL	542
10.7	Future Directions and Perspectives	546
10.7.1	Distributed Clocking Using DLLs	546
10.7.2	Optical Clock Distribution	548
10.7.3	Synchronous versus Asynchronous Design	549
10.8	Summary	550
10.9	To Probe Further	551
	References	551
Design Methodology Insert G	Design Verification	553
	References	557

Chapter 11	Designing Arithmetic Building Blocks	559
11.1	Introduction	560
11.2	Datapaths in Digital Processor Architectures	560
11.3	The Adder	561
11.3.1	The Binary Adder: Definitions	561
11.3.2	The Full Adder: Circuit Design Considerations	564
11.3.3	The Binary Adder: Logic Design Considerations	571
11.4	The Multiplier	586
11.4.1	The Multiplier: Definitions	586
11.4.2	Partial-Product Generation	587
11.4.3	Partial-Product Accumulation	589
11.4.4	Final Addition	593
11.4.5	Multiplier Summary	594
11.5	The Shifter	594
11.5.1	Barrel Shifter	595
11.5.2	Logarithmic Shifter	596
11.6	Other Arithmetic Operators	596
11.7	Power and Speed Trade-offs in Datapath Structures*	600
11.7.1	Design Time Power-Reduction Techniques	601
11.7.2	Run-Time Power Management	611
11.7.3	Reducing the Power in Standby (or Sleep) Mode	617
11.8	Perspective: Design as a Trade-off	618
11.9	Summary	619
11.10	To Probe Further	620
	References	621
Chapter 12	Designing Memory and Array Structures	623
12.1	Introduction	624
12.1.1	Memory Classification	625
12.1.2	Memory Architectures and Building Blocks	627
12.2	The Memory Core	634
12.2.1	Read-Only Memories	634
12.2.2	Nonvolatile Read-Write Memories	647
12.2.3	Read-Write Memories (RAM)	657
12.2.4	Contents-Addressable or Associative Memory (CAM)	670
12.3	Memory Peripheral Circuitry*	672
12.3.1	The Address Decoders	672
12.3.2	Sense Amplifiers	679
12.3.3	Voltage References	686
12.3.4	Drivers/Buffers	689
12.3.5	Timing and Control	689

12.4	Memory Reliability and Yield*	693
12.4.1	Signal-to-Noise Ratio	693
12.4.2	Memory Yield	698
12.5	Power Dissipation in Memories*	701
12.5.1	Sources of Power Dissipation in Memories	701
12.5.2	Partitioning of the Memory	702
12.5.3	Addressing the Active Power Dissipation	702
12.5.4	Data-Retention Dissipation	704
12.5.5	Summary	707
12.6	Case Studies in Memory Design	707
12.6.1	The Programmable Logic Array (PLA)	707
12.6.2	A 4-Mbit SRAM	710
12.6.3	A 1-Gbit NAND Flash Memory	712
12.7	Perspective: Semiconductor Memory Trends and Evolutions	714
12.8	Summary	716
12.9	To Probe Further	717
	References	718
<i>Design Methodology Insert H Validation and Test</i>		
of Manufactured Circuits		721
H.1	Introduction	721
H.2	Test Procedure	722
H.3	Design for Testability	723
H.3.1	Issues in Design for Testability	723
H.3.2	Ad Hoc Testing	725
H.3.3	Scan-Based Test	726
H.3.4	Boundary-Scan Design	729
H.3.5	Built-in Self-Test (BIST)	730
H.4	Test-Pattern Generation	734
H.4.1	Fault Models	734
H.4.2	Automatic Test-Pattern Generation (ATPG)	736
H.4.3	Fault Simulation	737
H.5	To Probe Further	737
	References	737
Problem Solutions		739
Index		745

PART

1

The Fabrics

"The complexity for minimum component costs has increased at a rate of roughly a factor of two per year. Certainly over the short term, this rate can be expected to continue, if not to increase. Over the longer term, the rate of increase is a bit more uncertain, although there is no reason to believe it will not remain nearly constant for at least 10 years. That means by 1975, the number of components per integrated circuit for minimum cost will be 65,000."

Gordon Moore,
Cramming more Components onto
Integrated Circuits, (1965).



CHAPTER

1

Introduction

The evolution of digital circuit design

Compelling issues in digital circuit design

How to measure the quality of a design

Valuable references

- 1.1 A Historical Perspective
- 1.2 Issues in Digital Integrated Circuit Design
- 1.3 Quality Metrics of a Digital Design
 - 1.3.1 Cost of an Integrated Circuit
 - 1.3.2 Functionality and Robustness
 - 1.3.3 Performance
 - 1.3.4 Power and Energy Consumption
- 1.4 Summary
- 1.5 To Probe Further

1.1 A Historical Perspective

The concept of digital data manipulation has made a dramatic impact on our society. One has long grown accustomed to the idea of digital computers. Evolving steadily from mainframe and minicomputers, personal and laptop computers have proliferated into daily life. More significant, however, is a continuous trend towards digital solutions in all other areas of electronics. Instrumentation was one of the first noncomputing domains where the potential benefits of digital data manipulation over analog processing were recognized. Other areas such as control were soon to follow. Only recently have we witnessed the conversion of telecommunications and consumer electronics into the digital format. Increasingly, telephone data is transmitted and processed digitally over both wired and wireless networks. The compact disk has revolutionized the audio world, and digital video is following in its footsteps.

The idea of implementing computational engines using an encoded data format is by no means an idea of our times. In the early 19th century, Babbage envisioned large-scale mechanical computing devices, which he called *Difference Engines* [Swade93]. Although these engines use the decimal number system rather than the binary representation now common in modern electronics, the underlying concepts are very similar. The Analytical Engine, developed in 1834, was perceived as a general-purpose computing machine, with features strikingly close to modern computers. Besides executing the basic repertoire of operations (addition, subtraction, multiplication, and division) in arbitrary sequences, the machine operated in a two-cycle sequence, called “store” and “mill” (execute), not unlike today’s computers. It even used pipelining to speed up the execution of the addition operation! Unfortunately, the complexity and the cost of the designs made the concept impractical. For instance, the design of Difference Engine I (part of which is shown in Figure 1-1) required 25,000 mechanical parts at a total cost of £17,470 (in 1834!).

The electrical solution turned out to be more cost effective. Early digital electronics systems were based on magnetically controlled switches (or relays). They were mainly used in the implementation of very simple logic networks. Such systems are still used in train safety systems. The age of digital electronic computing only started in full with the introduction of the vacuum tube. While originally used almost exclusively for analog processing, it soon was recognized that the vacuum tube was useful for digital computations as well. Not long thereafter, the first complete computers were realized. The era of the vacuum-tube-based computer culminated in the design of machines like the ENIAC (intended for computing artillery firing tables) and the UNIVAC I (the first successful commercial computer). To get an idea about *integration density*, the ENIAC was 80 feet long, 8.5 feet high, and several feet wide; it also incorporated 18,000 vacuum tubes. It became rapidly clear, however, that this design technology had reached its limits. Reliability problems and excessive power consumption made the implementation of larger engines economically and practically infeasible.

All changed with the invention of the *transistor* at Bell Telephone Laboratories in 1947 [Bardeen48], followed by the introduction of the bipolar junction transistor by Shockley in

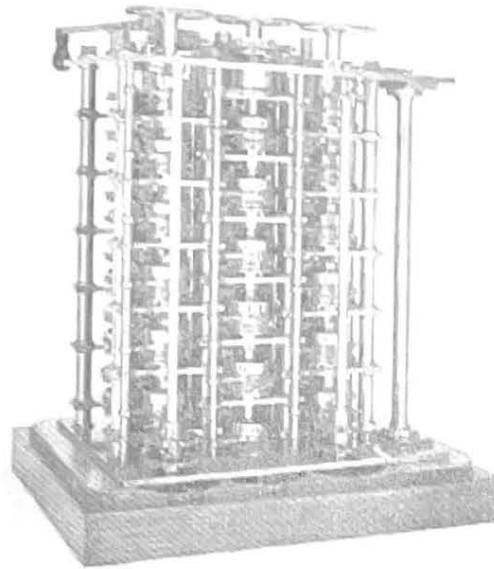


Figure 1-1 Working part of Babbage's Difference Engine I (1832), the first known automatic calculator (from [Swade93], courtesy of the Science Museum of London).

1949 [Schockley49].¹ It took until 1956 before this led to the first bipolar digital logic gate, made of discrete components introduced by Harris [Harris56]. In 1958, Jack Kilby at Texas Instruments conceived the *integrated circuit (IC)*, in which all components, passive and active, are integrated on a single semiconductor substrate—he was awarded the Nobel prize for this breakthrough. It led to the introduction of the first set of integrated-circuit commercial logic gates, called the *Fairchild Micrologic family* [Norman60]. The first truly successful IC logic family, *TTL (Transistor-Transistor Logic)* was pioneered in 1962 [Beeson62]. Other logic families were devised with higher performance in mind. Examples of these are the current switching circuits that produced the first subnanosecond digital gates and culminated in the *ECL (Emitter-Coupled Logic)* family [Masaki74]. TTL had the advantage, however, of offering a higher integration density and was the basis of the first integrated circuit revolution. In fact, the manufacturing of TTL components is what spearheaded the first large semiconductor companies such as Fairchild, National, and Texas Instruments. The family was so successful that it composed the largest fraction of the digital semiconductor market until the 1980s.

Ultimately, bipolar digital logic lost the battle for hegemony in the digital design world for exactly the reasons that haunted the vacuum tube approach: the large power consumption per gate puts an upper limit on the number of gates that can be reliably integrated on a single die,

¹An intriguing overview of the evolution of digital integrated circuits can be found in [Murphy93]. (Most of the data in this overview has been extracted from this reference). It is accompanied by some of the historically ground-breaking publications in the domain of digital ICs.

package, housing, or box. Although attempts were made to develop high integration density, low-power bipolar families (such as I^2L —*Integrated Injection Logic* [Hart72]), the MOS digital integrated circuit approach eventually held sway.

The basic principle behind the MOSFET transistor (originally called IGFET) was proposed in a patent by J. Lilienfeld (Canada) as early as 1925, and, independently, by O. Heil in England in 1935. Insufficient knowledge of the materials and gate stability problems, however, delayed the practical usability of the device for a long time. Once these were solved, MOS digital integrated circuits started to take off in full in the early 1970s. Remarkably, the first MOS logic gates introduced were of the CMOS variety [Wanlass63], and this trend continued until the late 1960s. The complexity of the manufacturing process delayed the full exploitation of these devices for two more decades. Instead, the first practical MOS integrated circuits were implemented in PMOS-only logic and were used in applications such as calculators. The second age of the digital integrated circuit revolution was inaugurated with the introduction of the first microprocessors by Intel in 1972 (the 4004) [Faggin72] and 1974 (the 8080) [Shima74]. These processors were implemented in NMOS-only logic, which has the advantage of higher speed over the PMOS logic. Simultaneously, MOS technology enabled the realization of the first high-density semiconductor memories. For instance, the first 4Kbit MOS memory was introduced in 1970 [Hoff70].

These events were at the start of a truly astounding evolution towards ever higher integration densities and speed performances, a revolution that currently is still in full swing. The road to the current levels of integration has not been without hindrances, however. In the late 1970s, NMOS-only logic started to suffer from the same plague that made high-density bipolar logic unattractive or infeasible: power consumption. This realization, combined with progress in manufacturing technology, finally tilted the balance towards the CMOS technology, and this is where it remains today. Interestingly enough, power consumption concerns are rapidly becoming dominant in CMOS design as well, and this time there does not seem to be a new technology around the corner to alleviate the problem.

Although the large majority of the current integrated circuits are implemented in the MOS technology, other technologies come into play when very high performance is at stake. An example of this is the BiCMOS technology that combines bipolar and MOS devices on the same die. When even higher performance is necessary, other technologies emerge such as silicon-germanium, and even superconducting technologies. These technologies only play a very small role in the overall digital integrated circuit design scene. With the ever increasing performance of CMOS, this role is bound to be reduced further with time—hence, the focus of this textbook is on CMOS only.

1.2 Issues in Digital Integrated Circuit Design

Integration density and performance of integrated circuits have gone through an astounding revolution in the last two decades. In the 1960s, Gordon Moore, then with Fairchild Corporation and later cofounder of Intel, predicted that the number of transistors that can be integrated on a

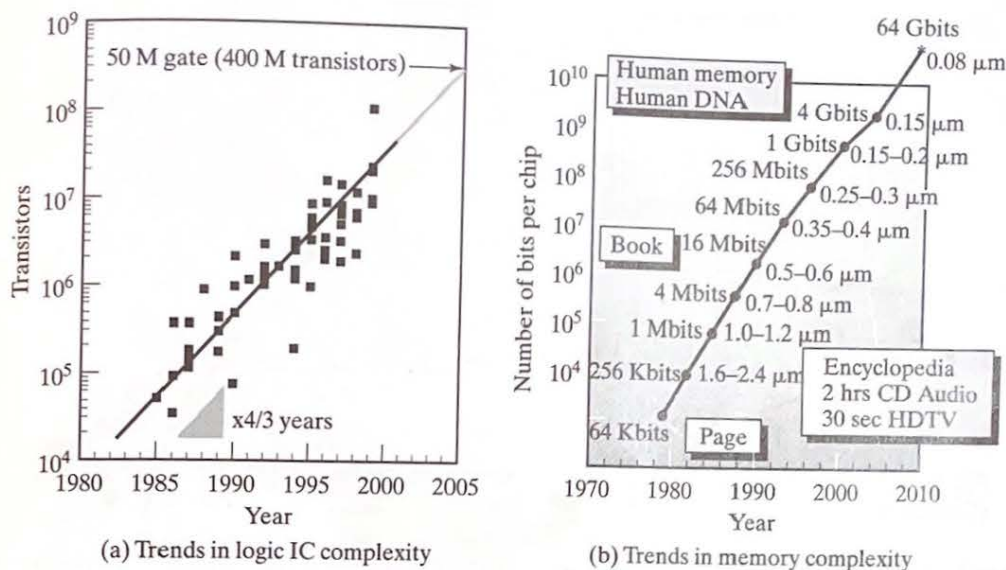


Figure 1-2 Evolution of integration complexity of logic ICs and memories as a function of time.

single die would grow exponentially with time. This prediction, later called *Moore's law*, has proven to be amazingly visionary [Moore65]. Its validity is best illustrated with the aid of a set of graphs. Figure 1-2 plots the integration density of both logic ICs and memory as a function of time. As can be observed, integration complexity doubles approximately every one to two years. As a result, memory density has increased by more than a thousandfold since 1970.

An intriguing case study is offered by the microprocessor. From its inception in the early 1970s, the microprocessor has grown in performance and complexity at a steady and predictable pace. The transistor counts for a number of landmark designs that are collected in Figure 1-3.

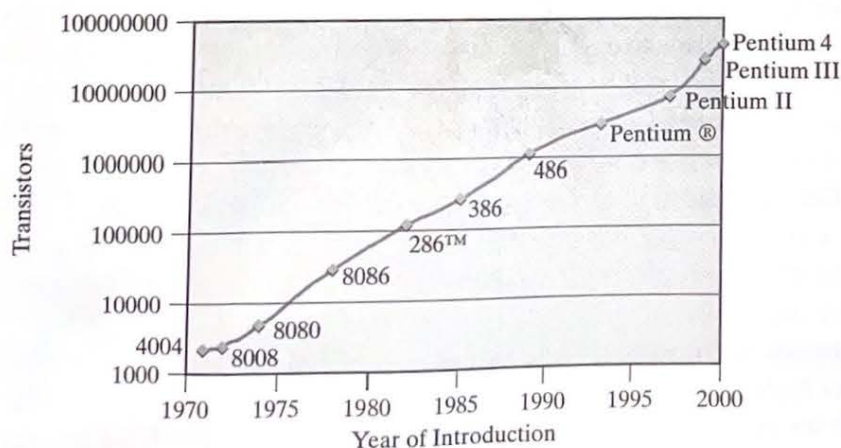


Figure 1-3 Historical evolution of microprocessor transistor count (from [Intel01]).

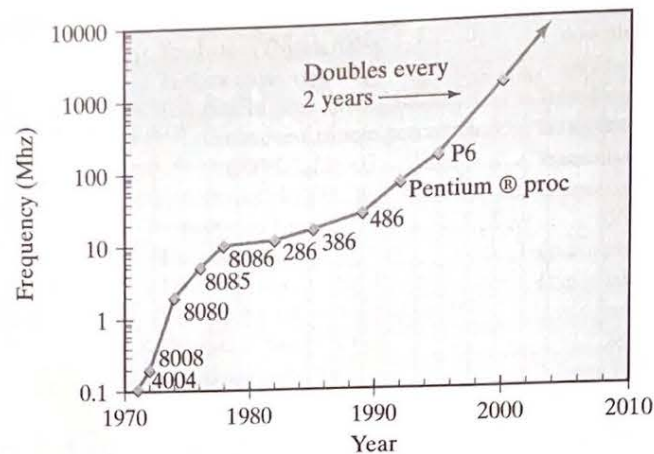


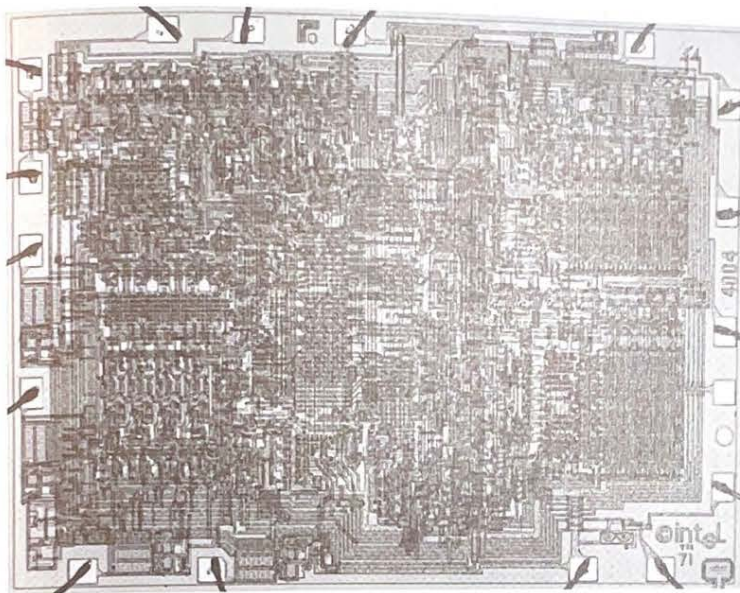
Figure 1-4 Microprocessor performance trends at the beginning of the 21st century. (Courtesy of Intel.)

The million-transistor/chip barrier was crossed in the late 1980s. Clock frequencies double every three years in the past decade and have reached into the GHz range. This is illustrated in Figure 1-4, which plots the microprocessor trends in terms of performance at the beginning of the 21st century. An important observation is that, as of now, these trends have not shown any signs of a slowdown.

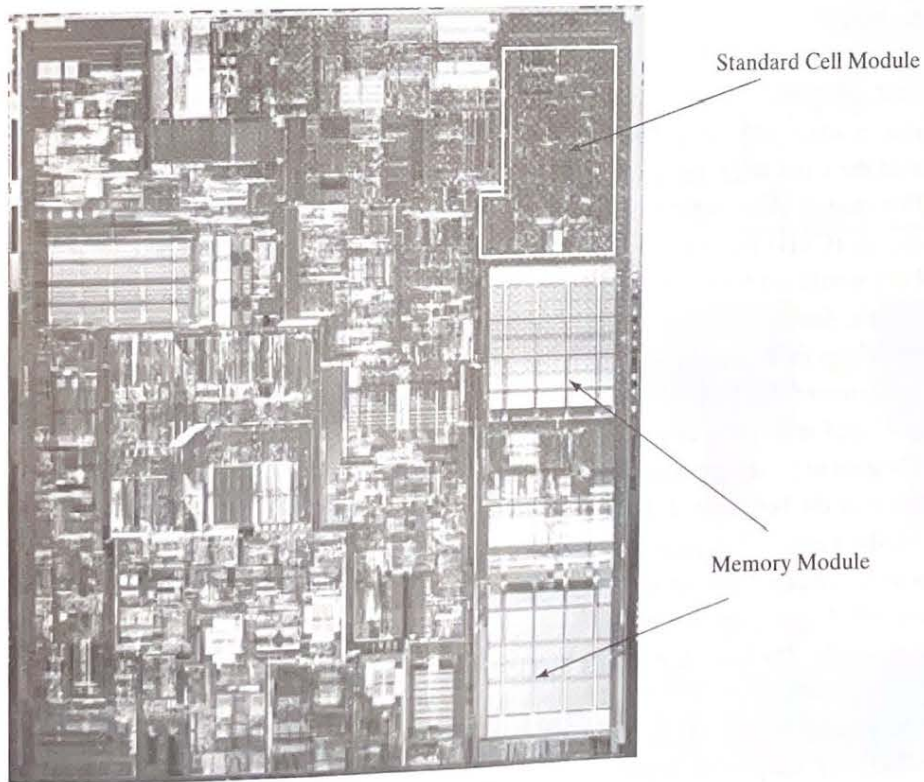
It should not surprise the reader that this revolution has had a profound impact on how digital circuits are designed. Early designs were truly handcrafted. Every transistor was laid out and optimized individually and carefully fitted into its environment. This is adequately illustrated in Figure 1-5a, which shows the design of the Intel 4004 microprocessor. Obviously, this approach is not appropriate when more than a million devices have to be created and assembled. With the rapid evolution of the design technology, time to market is one of the crucial factors in the ultimate success of a component.

As a result, designers have increasingly adhered to rigid design methodologies and strategies that are more amenable to design automation. The impact of this approach is apparent from the layout of one of the later Intel microprocessors, the Pentium® 4, shown in Figure 1-5b. Instead of the individualized approach of the earlier designs, a circuit is constructed in a hierarchical way: a processor is a collection of modules, each of which consists of a number of cells on its own. Cells are reused as much as possible to reduce the design effort and to enhance the chances for a first-time-right implementation. The fact that this hierarchical approach is at all possible is the key ingredient for the success of digital circuit design and also explains why, for instance, very large-scale analog design has never caught on.

The obvious next question is why such an approach is feasible in the digital world and not (or to a lesser degree) in analog designs. The crucial concept here, and the most important one in dealing with the complexity issue, is *abstraction*. At each design level, the internal details of a complex module can be abstracted away and replaced by a *black-box view* or *model*. This model contains virtually all the information needed to deal with the block at the next level of hierarchy.



(a) The 4004 microprocessor



(b) The Pentium® 4 microprocessor

Figure 1-5 Comparing the design methodologies of the Intel 4004 (1971) and Pentium® 4 (2000) microprocessors (reprinted with permission from Intel).

For instance, once a designer has implemented a multiplier module, its performance can be defined very accurately and can be captured in a model. In general, the performance of this multiplier is only marginally influenced by the way it is utilized in a larger system. For all purposes, therefore, it can be considered a black box with known characteristics. As there exists no compelling need for the system designer to look inside this box, design complexity is substantially reduced. The impact of this *divide-and-conquer* approach is dramatic. Instead of having to deal with a myriad of elements, the designer has to consider only a handful of components, each of which are characterized in performance and cost by a small number of parameters.

This is analogous to a software designer using a library of software routines such as input/output drivers. Someone writing a large program does not bother to look inside those library routines. The only thing he cares about is the intended result of calling one of those modules. Imagine what writing software programs would be like if you had to fetch every bit individually from the disk and ensure its correctness instead of relying on handy “file open” and “get string” operators.

Abstraction levels typically used in digital circuit design are, in order of increasing abstraction, the device, circuit, gate, functional module (e.g., adder) and system levels (e.g., processor), as illustrated in Figure 1-6. A semiconductor device is an entity with a very complex behavior. No circuit designer will ever seriously consider the solid-state physics equations governing the behavior of the device when designing a digital gate. Instead, he will use a simplified model that adequately describes the input/output behavior of the transistor. For instance, an AND gate is adequately described by its Boolean expression ($Z = A.B$), its bounding box, the position of the input and output terminals, and the delay between the inputs and the output.

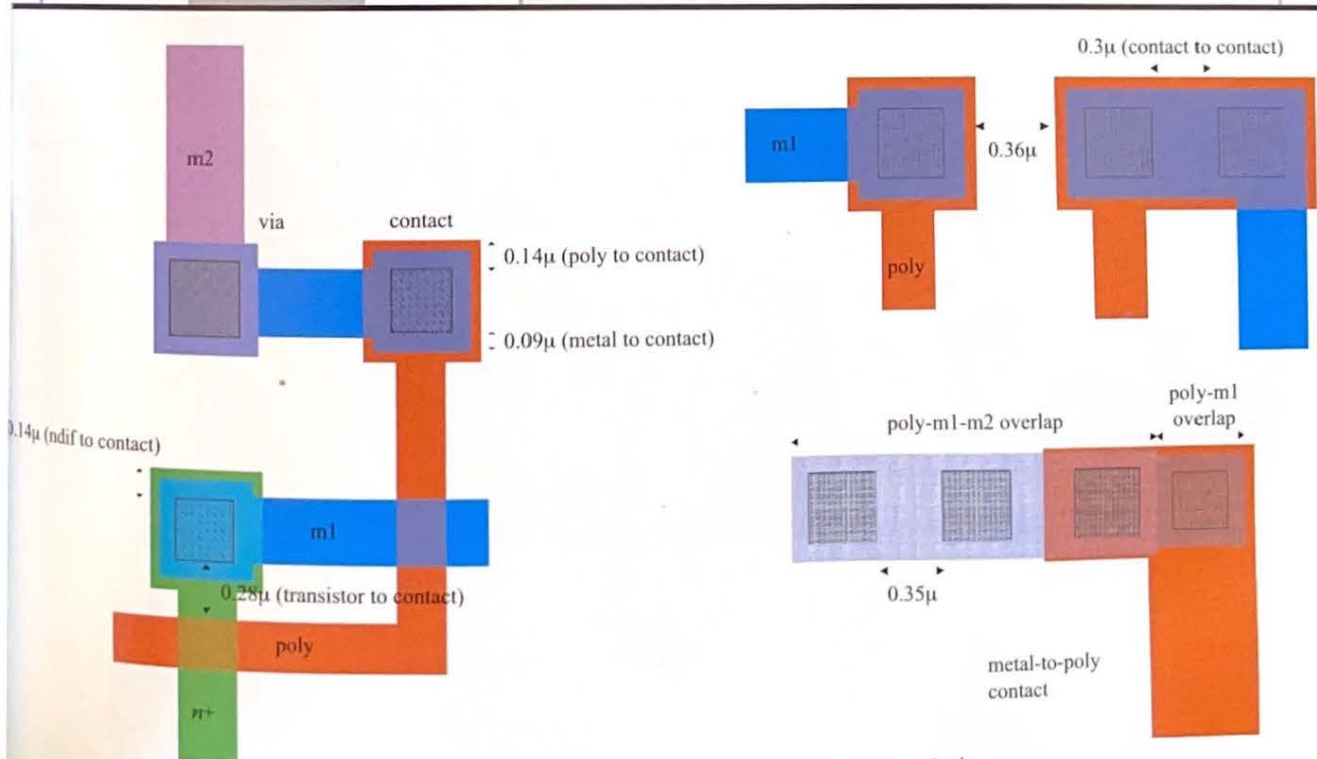
This design philosophy has been the enabler for the emergence of elaborate *computer-aided design* (CAD) frameworks for digital integrated circuits—without it the current design complexity would not have been achievable. Design tools include simulation at the various complexity levels, design verification, layout generation, and design synthesis. An overview of these tools and design methodologies is given in Chapter 8.

Furthermore, to avoid the redesign and reverification of frequently used cells, such as basic gates and arithmetic and memory modules, designers most often resort to *cell libraries*. These libraries not only contain the layouts, but also provide complete documentation and characterization of the behavior of the cells. The use of cell libraries is, for instance, apparent in the layout of the Pentium® 4 processor (Figure 1-5b). The integer and floating-point unit, just to name a few, contain large sections designed using one particular cell-based approach, called *standard cell*. Logic gates are placed in rows of cells of equal height and interconnected using routing channels. The layout of such a block can be generated automatically given that a library of cells is available.

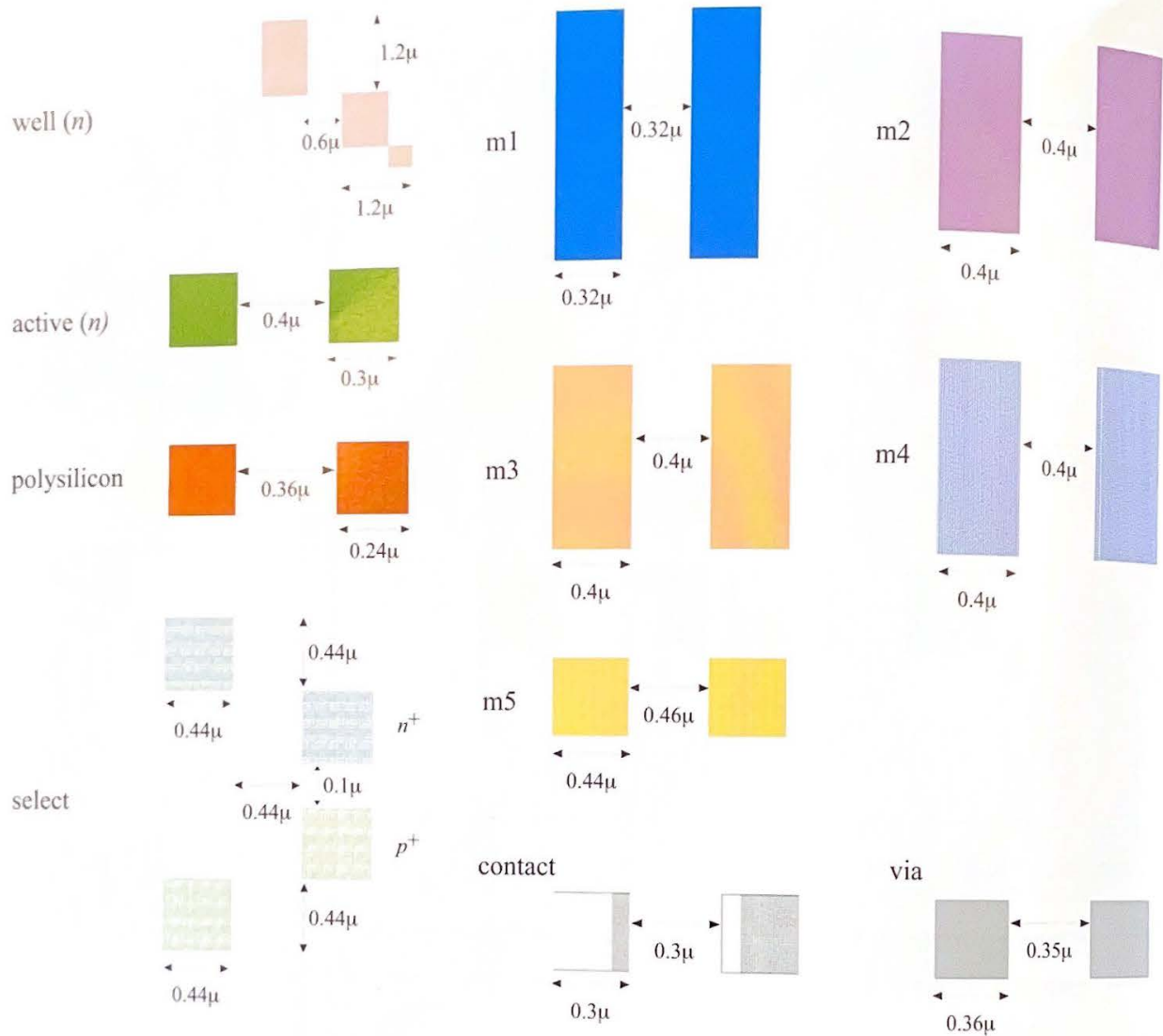
The preceding analysis demonstrates that design automation and modular design practices have effectively addressed some of the complexity issues incurred in contemporary digital design. This leads to the following pertinent question: If design automation solves all our design problems, why should we be concerned with digital circuit design at all? Will the next-generation

Layer Description	Representation				
metal					
	m1	m2	m3	m4	m5
well					
	nw				
polysilicon					
	poly				
contacts & vias					
	ct	v12,v23,v34,v45	nwc	pwc	
active area and FETs					
	ndif	pdif	nfet	pfet	
select					
	nplus	pplus	prb		

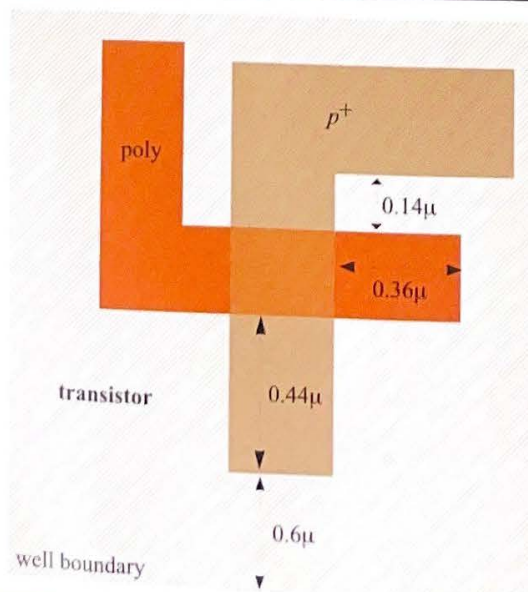
Colorplate 1. CMOS layers and representations
(for vanilla 0.25 μm CMOS process)



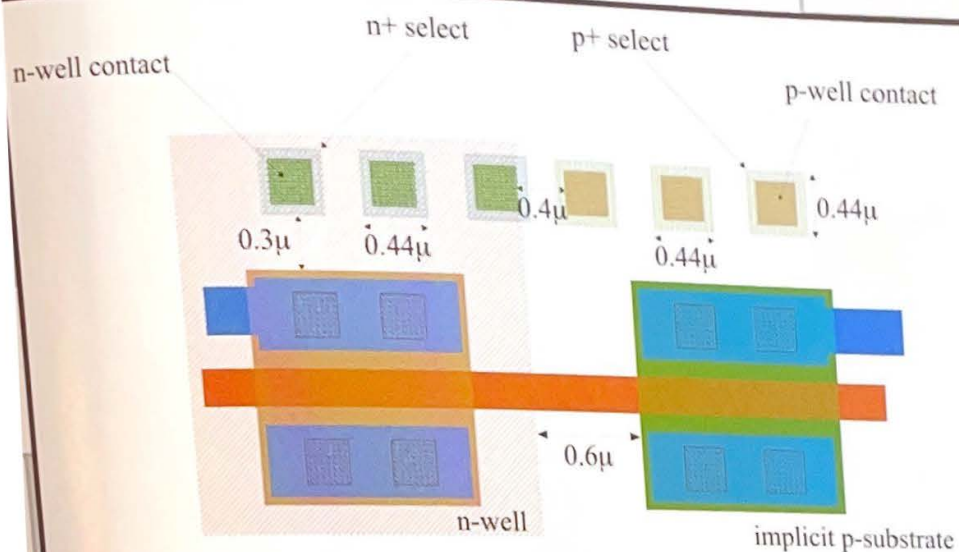
Colorplate 4. Design rules regarding contacts and vias.
Overlapping layers are marked by merged colors.



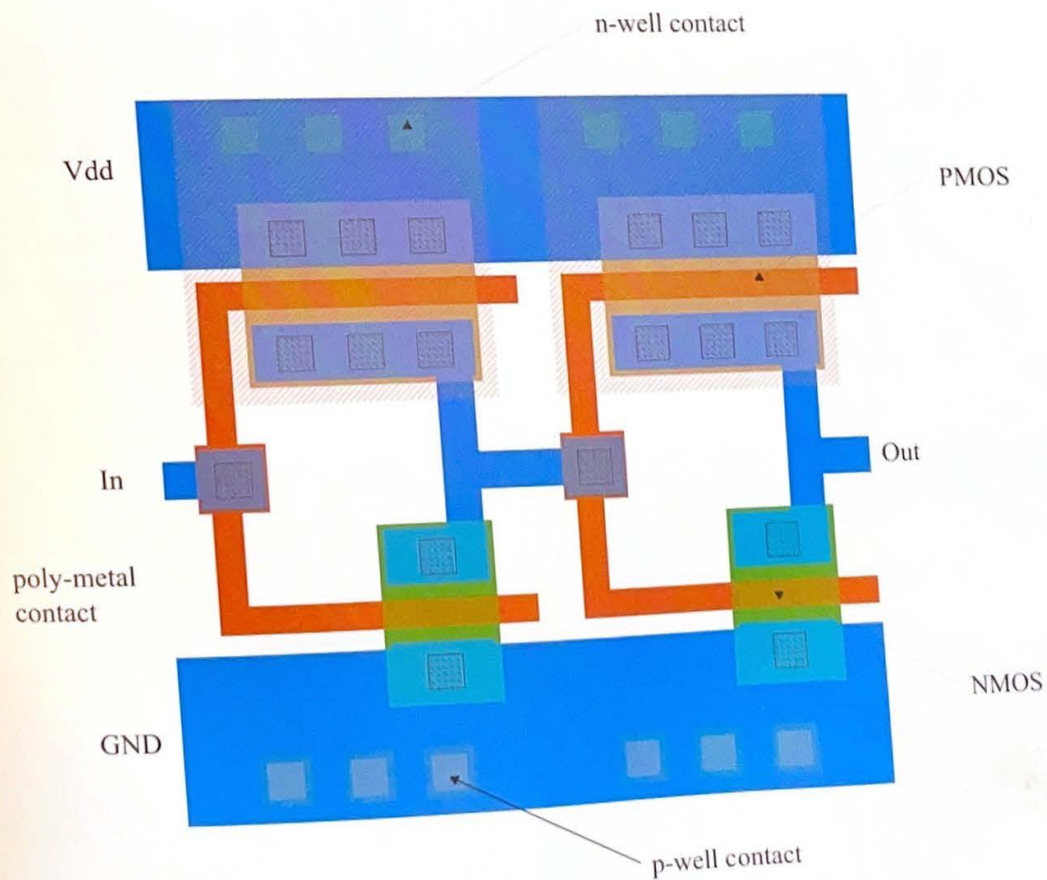
Colorplate 2. Intra-layer layout design rules, expressed as minimum dimensions and spacings.



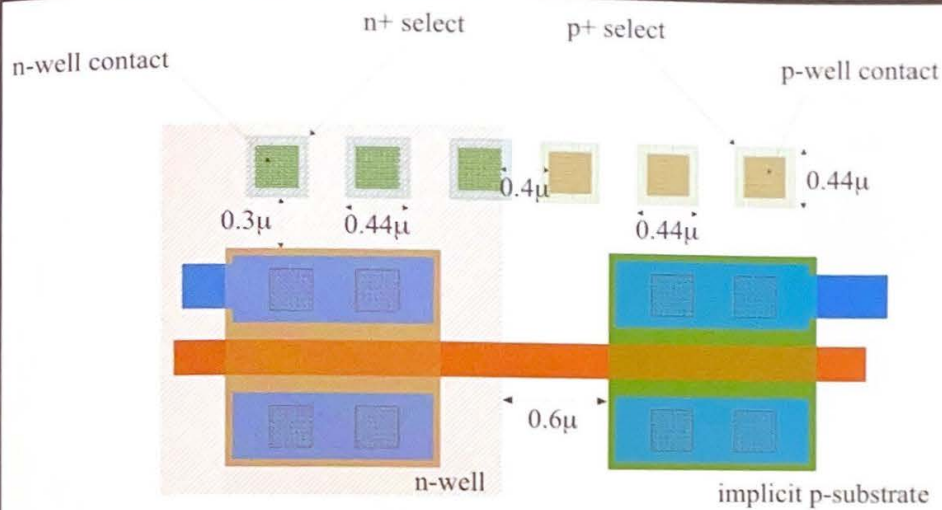
Colorplate 3. Design rules concerning transistor layout. The device shown is a PMOS transistor.



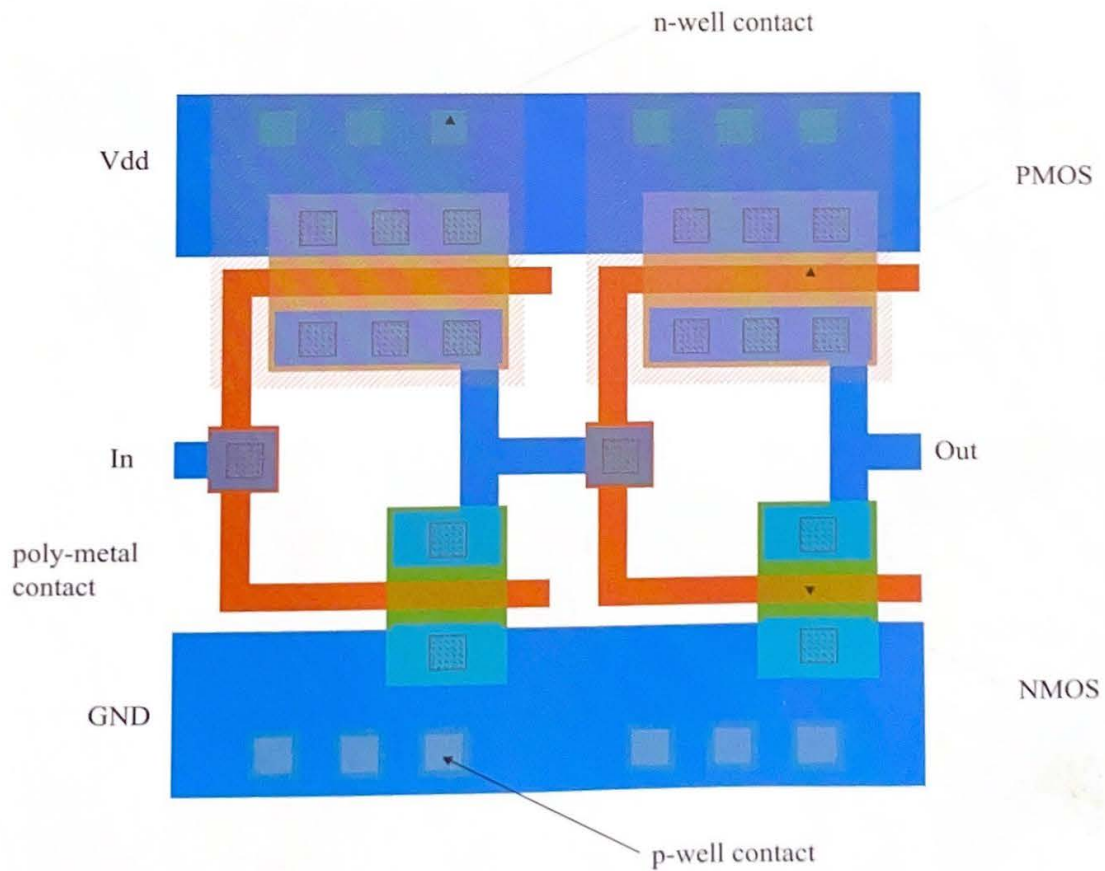
Colorplate 5. Design rules regarding well contacts and select layers.



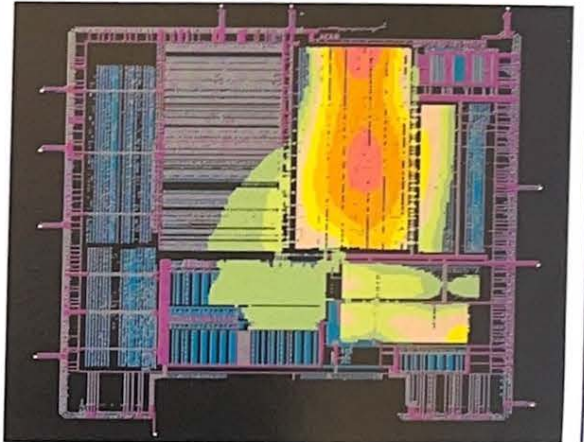
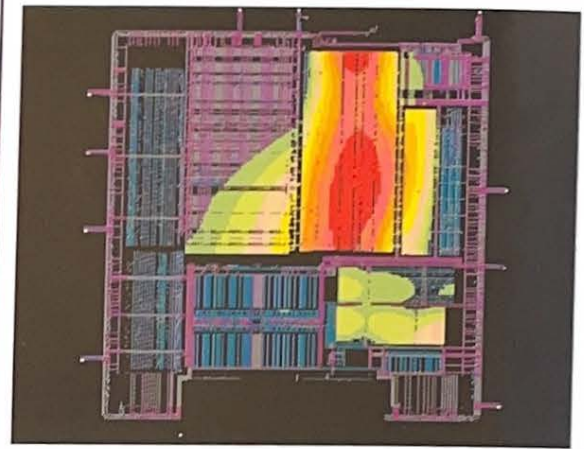
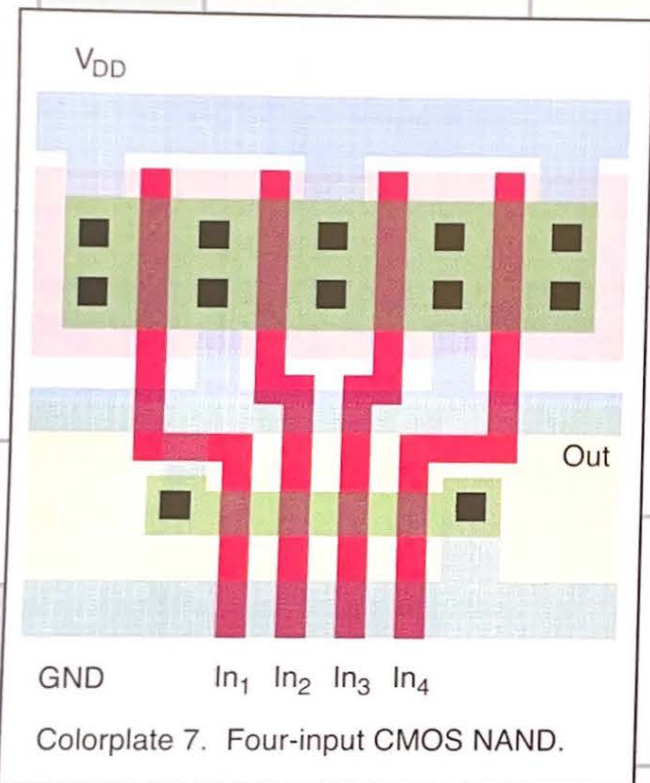
Colorplate 6. Layout of inverter in 0.25 μ m CMOS technology.



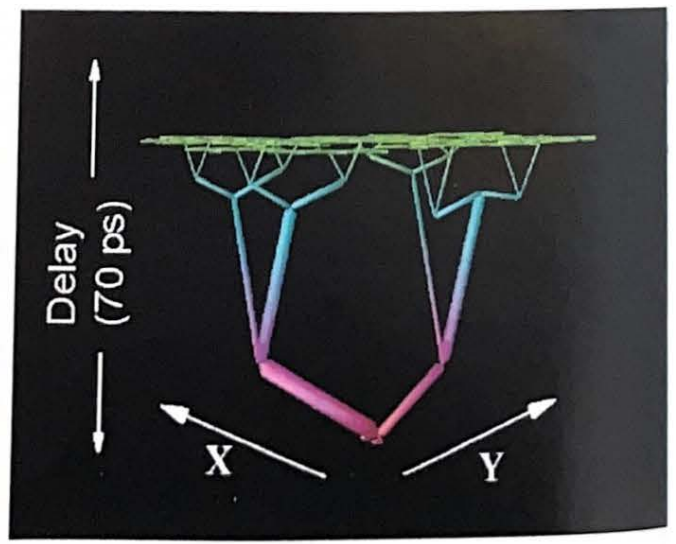
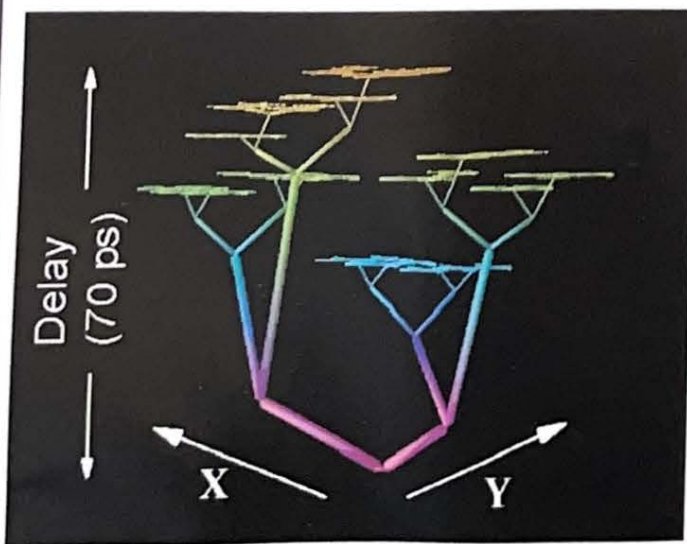
Colorplate 5. Design rules regarding well contacts and select layers.



Colorplate 6. Layout of inverter in 0.25 μm CMOS technology.



Colorplate 8. Simulated IR voltage drop in power-distribution network.



Colorplate 9. Visualizations of clock delay in a tree network driving different loads.

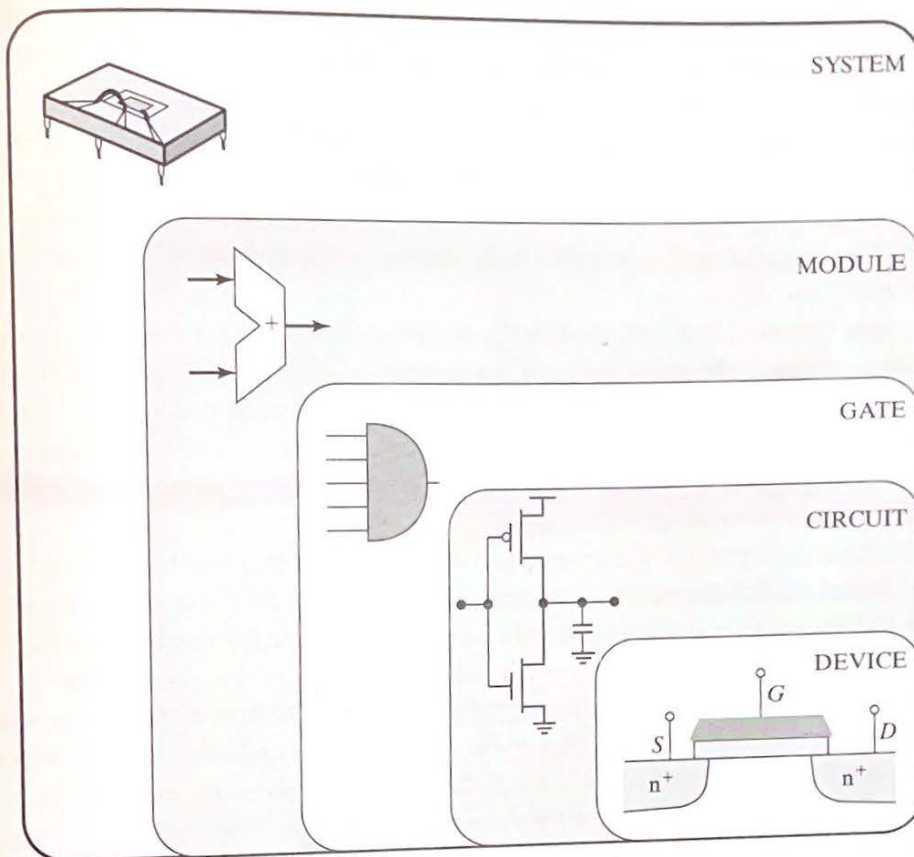


Figure 1-6 Design abstraction levels in digital circuits.

digital designer ever have to worry about transistors or parasitics, or is the smallest design entity he will ever consider be the gate and the module?

The truth is that the reality is more complex, and various reasons exist as to why an insight into digital circuits and their intricacies will still be an important asset for a long time to come:

- First, someone still has to *design and implement* the module libraries. Semiconductor technologies continue to advance from year to year. Until one has developed a foolproof approach towards “porting” a cell from one technology to another, each change in technology—which happens approximately every two years—requires a redesign of the library.
- Creating an adequate *model* of a cell or module requires an in-depth understanding of its internal operation. For instance, to identify the dominant performance parameters of a given design, one has to recognize the critical timing path first.
- The library-based approach works fine when the design constraints (speed, cost, or power) are not stringent. This is the case for a large number of *application-specific designs*, where

the main goal is to provide a more integrated system solution, and performance requirements are easily within the capabilities of the technology. Unfortunately, for a large number of other products such as microprocessors, success hinges on high performance, and designers therefore tend to push technology to its limits. At that point, the hierarchical approach tends to become somewhat less attractive. To resort to our previous analogy to software methodologies, a programmer tends to “customize” software routines when execution speed is crucial; compilers—or design tools—are not yet to the level of what human sweat or ingenuity can deliver.

- Even more important is the observation that the abstraction-based approach is only correct to a certain degree. The performance of, for instance, an adder can be substantially influenced by the way it is connected to its environment. The interconnection wires themselves contribute to delay because they introduce parasitic capacitances, resistances, and even inductances. **The impact of the interconnect parasitics is bound to increase in the years to come with the scaling of the technology.**
- Scaling tends to emphasize some other deficiencies of the abstraction-based model. Some design entities tend to be *global or external*. Examples of global factors are the clock signals, used for synchronization in a digital design, and the supply lines. Increasing the size of a digital design has a profound effect on these global signals. For instance, connecting more cells to a supply line can cause a voltage drop over the wire, which, in turn, can slow down all the connected cells. Issues such as clock distribution, circuit synchronization, and supply-voltage distribution are becoming more and more critical. Coping with them requires a profound understanding of the intricacies of digital circuit design.
- Another impact of technology evolution is that *new design issues* and constraints tend to emerge over time. A typical example of this is the periodical reemergence of power dissipation as a constraining factor, as was already illustrated in the historical overview. Another example is the changing ratio between device and interconnect parasitics. To cope with these unforeseen factors, one must at least be able to model and analyze their impact, requiring once again a profound insight into circuit topology and behavior.
- Finally, when things can go wrong, they often do. A fabricated circuit does not always exhibit the exact waveforms one might expect from advance simulations. Deviations can be caused by variations in the fabrication process parameters, or by the inductance of the package, or by a badly modeled clock signal. *Troubleshooting* a design requires circuit expertise.

For all these reasons, it is our profound belief that an in-depth knowledge of digital circuit design techniques and approaches is an essential asset for a digital-system designer. Even though she might not have to deal with the details of the circuit on a daily basis, the understanding will help her cope with unexpected circumstances and determine the dominant effects when analyzing a design.

Example 1.1 Clocks Defy Hierarchy

To illustrate some of the issues raised in the preceding discussion, let us examine the impact of deficiencies in one of the most important global signals in a design, the *clock*. The function of the clock signal in a digital design is to order the multitude of events happening in the circuit. This task can be compared to the function of a traffic light that determines which cars are allowed to move. It also makes sure that all operations are completed before the next one starts—a traffic light should be green long enough to allow a car or a pedestrian to cross the road. Under ideal circumstances, the clock signal is a periodic step waveform with transitions synchronized throughout the designed circuit (Figure 1-7a). In light of our analogy, changes in the traffic lights should be synchronized to maximize throughput while avoiding accidents. The importance of the *clock alignment* concept is illustrated with the example of two cascaded registers, both operating on the rising edge of the clock ϕ (Figure 1-7b). Under normal operating conditions, the input *In* gets sampled into the first register on the rising edge of ϕ and appears at the output exactly one clock period later. This is confirmed by the simulations shown in Figure 1-7c (signal *Out*).

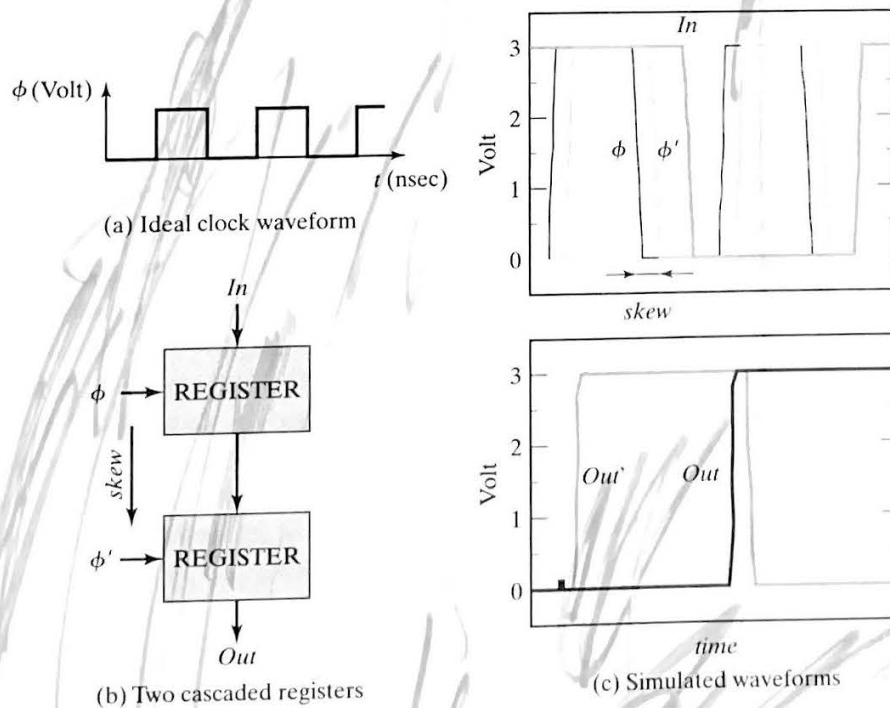


Figure 1-7 Impact of clock misalignment.

Due to delays associated with routing the clock wires, it may happen that the clocks become misaligned with respect to each other. As a result, the registers are interpreting time indicated by the clock signal differently. Consider the case in which the clock signal

for the second register is delayed—or skewed—by a value δ . The rising edge of the delayed clock ϕ' will postpone the sampling of the input of the second register. If the time it takes to propagate the output of the first register to the input of the second register is smaller than the clock delay, the latter will sample the wrong value. This causes the output to change prematurely, as clearly illustrated in the simulation, where the signal *Out'* goes high at the first rising edge of ϕ' instead of the second one. In terms of our traffic analogy, cars leaving the first traffic light hit the cars at the next light that have not yet left.

Clock misalignment, or *clock skew*, is an important example of how global signals may influence the functioning of a hierarchically designed system. Clock skew is actually one of the most critical design problems facing the designers of large high-performance systems.

Example 1.2 Power Distribution Networks Defy Hierarchy

While the clock signal is one example of a global signal that crosses the chip hierarchy boundaries, the power distribution network represents another. A digital system requires a stable DC voltage to be supplied to the individual gates. To ensure proper operation, this voltage should be stable within a few hundred millivolts. The power distribution system has to provide this stable voltage in the presence of very large current variations. The resistive nature of the on-chip wires and the inductance of the IC package pins make this a difficult proposition. For example, the average DC current to be supplied to a 100 W-1V microprocessor equals 100 A! The peak current can easily be twice as large, and current demand can readily change from almost zero to this peak value over a short time—in the range of 1 nsec or less. This leads to a current variation of 100 GA/s, which is a truly astounding number.

Consider the problem of the resistance of power distribution wires. For a current of 100 A, a wire resistance of merely 1.25 m Ω leads to a 5% drop in supply voltage (for a 2.5 V supply). Making the wires wider reduces the resistance and thus the voltage drop. While this sizing of the power network is relatively simple in a flat design approach, it is a lot more complex in a hierarchical design. For example, consider the two blocks shown in Figure 1-8a [Saleh01]. If power distribution for Block A is examined in isolation, the additional loading due to the presence of Block B is not taken into account. If power is routed through Block A to Block B, a larger IR drop will occur in Block B, since power is also being consumed by Block A before it reaches Block B.

Since the total IR drop is based on the resistance seen from the pin to the block, one could route around the block and feed power to each block separately, as shown in Figure 1-8b. Ideally, the main trunks should be large enough to handle all the current flowing through separate branches. Although routing power this way is easier to control and maintain, it also requires more area to implement. The large metal trunks of power have to

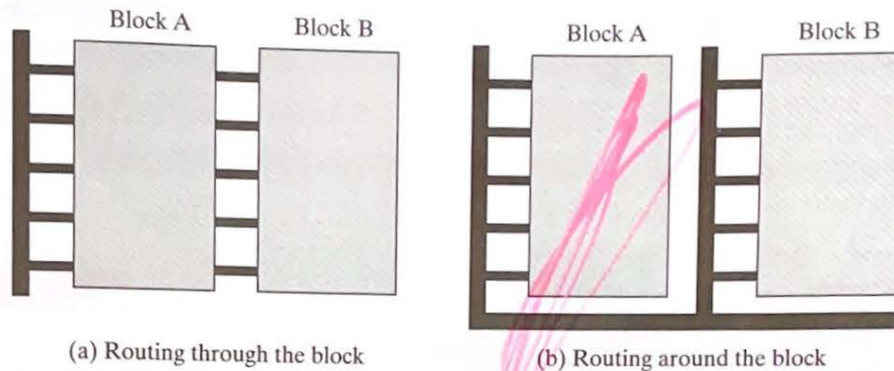


Figure 1-8 Power distribution network design.

be sized to handle all the current for each block. This requirement forces designers to set aside area for power busing that takes away from the available routing area.

As more and more blocks are added, the complex interactions between the blocks determine the actual voltage drops. For instance, it is not always easy to determine which way the current will flow when multiple parallel paths are available between the power source and the consuming gate. Also, currents into the different modules do rarely peak at the same time. All these considerations make the design of the power distribution a challenging job. It requires a design methodology approach that supersedes the artificial boundaries imposed by hierarchical design.

The purpose of this book is to provide *a bridge between the abstract vision of digital design and the underlying digital circuit and its peculiarities*. While starting from a solid understanding of the operation of electronic devices and an in-depth analysis of the nucleus of digital design—the inverter—we will gradually channel this knowledge into the design of more complex entities, such as complex gates, datapaths, registers, controllers, and memories. The persistent quest for a designer when designing each of the mentioned modules is to identify the dominant design parameters, to locate the section of the design he should focus his optimizations on, and to determine the specific properties that make the module under investigation (e.g., a memory) different from any others.

We also address other compelling (global) issues in modern digital circuit design such as *power dissipation, interconnect, timing, and synchronization*.

1.3 Quality Metrics of a Digital Design

This section defines a set of basic properties of a digital design. These properties help to quantify the quality of a design from different perspectives: cost, functionality, robustness, performance, and energy consumption. Which one of these metrics is most important depends upon the application. For instance, pure speed is a crucial property in a computer server. On the other hand, energy consumption is a dominant metric for hand held/mobile applications such as cell

phones. The introduced properties are relevant at all levels of the design hierarchy—system, chip, module, and gate. To ensure consistency in the definitions throughout the design hierarchy stack, we propose a bottom-up approach: We start with defining the basic quality metrics of a simple inverter, and gradually expand these to the more complex functions such as gate, module, and chip.

1.3.1 Cost of an Integrated Circuit

The total cost of any product can be separated into two components: the **recurring expenses** or the **variable cost**, and the **nonrecurring expenses** or the **fixed cost**.

Fixed Cost

The fixed cost is independent of the sales volume or the number of products sold. An important component of the fixed cost of an integrated circuit is the effort in time and manpower it takes to produce the design. This design cost is strongly influenced by the complexity of the design, the aggressiveness of the specifications, and the productivity of the designer. Advanced design methodologies that automate major parts of the design process can help to boost the latter. Bringing down the design cost in the presence of an ever-increasing IC complexity is one of the major challenges always facing the semiconductor industry.

Additionally, one has to account for the *indirect costs*, the company overhead that cannot be billed directly to one product. It includes the company's research and development (R&D) costs, manufacturing equipment, marketing and sales costs, and building infrastructure, among others.

Variable Cost

The variable cost accounts for the cost that is directly attributable to a manufactured product, and is hence proportional to the product volume. Variable costs include the costs of the parts used in the product, assembly costs, and testing costs. The total cost of an integrated circuit is

$$\text{cost per IC} = \text{variable cost per IC} + \left(\frac{\text{fixed cost}}{\text{volume}} \right) \quad (1.1)$$

The impact of the fixed cost is more pronounced for small-volume products. This helps explain why it makes sense to have large design teams working for several years on a hugely successful product such as a microprocessor.

While the cost of producing a single transistor has dropped exponentially over the past few decades, the basic variable-cost equation has not changed:

$$\text{variable cost} = \frac{\text{cost of die} + \text{cost of die test} + \text{cost of packaging}}{\text{final test yield}} \quad (1.2)$$

As will be discussed further in Chapter 2, the IC manufacturing process groups a number of identical circuits onto a single *wafer* (see Figure 1-9). Upon completion of the fabrication, the wafer is chopped into *dies*, which are then individually packaged after being *tested*. We will

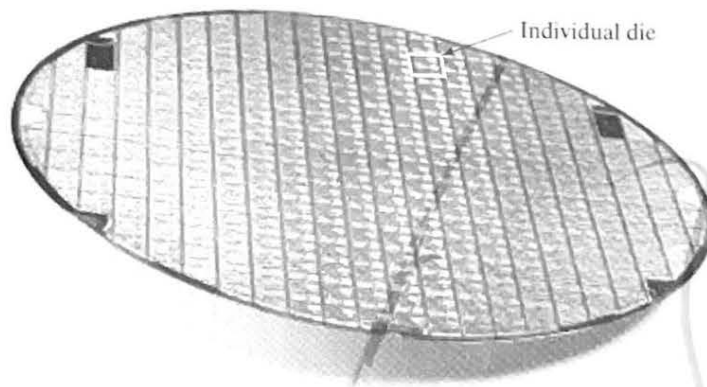


Figure 1-9 Finished wafer. Each square represents a die—in this case the AMD Duron™ microprocessor (reprinted with permission from AMD).

focus on the cost of the die in this discussion. The cost of packaging and testing is the topic of later chapters.

The die cost depends on the number of good dies on a wafer, and the percentage of those that are functional. The latter factor is called the *die yield*, given by

$$\text{cost of die} = \frac{\text{cost of wafer}}{\text{dies per wafer} \times \text{die yield}} \quad (1.3)$$

The number of dies per wafer is, in essence, the area of the wafer divided by the die area. The actual situation is somewhat more complicated as wafers are round, and chips are square. Dies around the perimeter of the wafer are therefore lost. The size of the wafer has been steadily increasing over the years, yielding more dies per fabrication run. Equation (1.3) also presents the first indication that the cost of a circuit is dependent upon the chip area—increasing the chip area simply means that fewer dies fit on a wafer.

The actual relation between cost and area is more complex, and depends upon the die yield. Both the substrate material and the manufacturing process introduce faults that can cause a chip to fail. Assuming that the defects are randomly distributed over the wafer and that the yield is inversely proportional to the complexity of the fabrication process, we obtain the equation

$$\text{die yield} = \left(1 + \frac{\text{defects per unit area} \times \text{die area}}{\alpha} \right)^{-\alpha} \quad (1.4)$$

where α is a parameter that depends upon the complexity of the manufacturing process, and it is roughly proportional to the number of masks. A good estimate for today's complex CMOS processes is $\alpha = 3$. The defects per unit area is a measure of the material and process-induced faults. A value between 0.5 and 1 defects/cm² is typical these days, but depends strongly upon the maturity of the process.

Example 1.3 Die Yield

Assume a wafer size of 12 inches, a die size of 2.5 cm^2 , 1 defects/ cm^2 , and $\alpha = 3$. Determine the die yield of this CMOS process run.

The number of dies per wafer can be estimated with the following expression, which takes into account the lost dies around the perimeter of the wafer:

$$\text{dies per wafer} = \frac{\pi \times (\text{wafer diameter}/2)^2}{\text{die area}} - \frac{\pi \times \text{wafer diameter}}{\sqrt{2 \times \text{die area}}}$$

This means there are 252 ($= 296 - 44$) potentially operational dies for this particular example. The die yield can be computed with the aid of Eq. (1.4), and equals 16%! This means that, on average only 40 of the dies will be fully functional.

The bottom line is that the number of functional dies per wafer, and hence the cost per die, is a strong function of the die area. While the yield tends to be excellent for the smaller designs, it drops rapidly once a certain threshold is exceeded. Bearing in mind the equations derived previously and the typical parameter values, we can conclude that die costs are proportional to the fourth power of the area:

$$\text{cost of die} = f(\text{die area})^4 \quad (1.5)$$

The area is a function that is directly controllable by the designer(s), and it is the prime metric for cost. Small area is thus a desirable property for a digital gate. The smaller the gate, the higher the integration density and the smaller the die size. Smaller gates also tend to be faster and consume less energy—the total gate capacitance, which is one of the dominant performance parameters, often scales with the area.

The number of transistors in a gate is indicative of the expected implementation area but other parameters may have an impact. For instance, a complex interconnect pattern between the transistors can cause the wiring area to dominate. The gate complexity, as expressed by the number of transistors and the regularity of the interconnect structure, also has an impact on the design cost. Complex structures are more difficult to implement and tend to take more of the designer's valuable time. Simplicity and regularity is a precious property in cost-sensitive designs.

1.3.2 Functionality and Robustness

A prime requirement for a digital circuit is, obviously, that it performs the function it is designed for. The measured behavior of a manufactured circuit normally deviates from the expected response. One reason for this aberration is due to the variations in the manufacturing process. The dimensions and device parameters vary between runs or even on a single wafer or die. The electrical behavior of a circuit can be profoundly affected by those variations. The presence of disturbing noise sources on or off the chip is another source of deviations in circuit response.

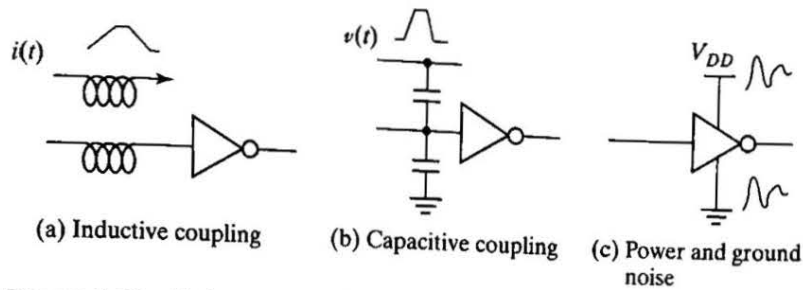


Figure 1-10 Noise sources in digital circuits.

The word *noise*, in the context of digital circuits, means *unwanted variations of voltages and currents at the logic nodes*. Noise signals can enter a circuit in many ways. Some examples of digital noise sources are depicted in Figure 1-10. For instance, two wires placed side by side in an integrated circuit form a coupling capacitor and a mutual inductance. Hence, a change in voltage or current on one of the wires can influence the signals on the neighboring wire. Noise on the power and ground rails of a gate also influences the signal levels in the gate.

Most noise in a digital system is internally generated, and the noise value is proportional to the signal swing. Capacitive and inductive cross talk, and the internally generated power supply noise are examples of such. Other noise sources such as input power supply noise are external to the system, and their value is not related to the signal levels. For these sources, the noise level is directly expressed in volts or amperes. Noise sources that are a function of the signal level are better expressed as a fraction or percentage of the signal level. Noise is a major concern in the engineering of digital circuits. How to cope with all these disturbances is one of the main challenges in the design of high-performance digital circuits and is a recurring topic in this book.

The steady-state parameters (also called the *static behavior*) of a gate measure how robust the circuit is with respect to both variations in the manufacturing process and noise disturbances. The definition and derivation of these parameters requires a prior understanding of how digital signals are represented in the world of electronic circuits.

Digital circuits (DC) perform operations on *logical* (or *Boolean*) variables. A logical variable x can only assume two discrete values:

$$x \in \{0,1\}$$

As an example, the inversion (i.e., the function that an inverter performs) implements the following compositional relationship between two Boolean variables x and y :

$$y = \bar{x}: \{x = 0 \Rightarrow y = 1; x = 1 \Rightarrow y = 0\} \quad (1.6)$$

A logical variable is, however, a mathematical abstraction. In a physical implementation, such a variable is represented by an electrical quantity. This is most often a node voltage that is not discrete, but can adopt a continuous range of values. This electrical voltage is turned into a discrete variable by associating a *nominal voltage level* with each logic state: $1 \Leftrightarrow V_{OH}$, $0 \Leftrightarrow V_{OL}$, where V_{OH} and V_{OL} represent the *high* and the *low* logic levels, respectively. Applying V_{OH}

to the input of an inverter yields V_{OL} at the output and vice versa. The difference between the two is called the *logic or signal swing* V_{sw} .

$$\begin{aligned} V_{OH} &= \overline{(V_{OL})} \\ V_{OL} &= \overline{(V_{OH})} \end{aligned} \quad (1.7)$$

The Voltage-Transfer Characteristic

Assume now that a logical variable *in* serves as the input to an inverting gate that produces the variable *out*. The electrical function of a gate is best expressed by its *voltage-transfer characteristic* (VTC) (sometimes called the *DC transfer characteristic*), which plots the output voltage as a function of the input voltage $V_{out} = f(V_{in})$. An example of an inverter VTC is shown in Figure 1-11. The high and low nominal voltages, V_{OH} and V_{OL} , can readily be identified— $V_{OH} = f(V_{OL})$ and $V_{OL} = f(V_{OH})$. Another point of interest of the VTC is the *gate or switching threshold voltage* V_M (not to be confused with the threshold voltage of a transistor), that is defined as $V_M = f(V_M)$. V_M can also be found graphically at the intersection of the VTC curve and the line given by $V_{out} = V_{in}$. The gate threshold voltage presents the midpoint of the switching characteristics, which is obtained when the output of a gate is short-circuited to the input. This point will prove to be of particular interest when studying circuits with feedback (also called *sequential circuits*).

Even if an ideal nominal value is applied at the input of a gate, the output signal often deviates from the expected nominal value. These deviations can be caused by noise or by the loading on the output of the gate (i.e., by the number of gates connected to the output signal). Figure 1-12a illustrates how a logic level is represented in reality by a range of acceptable voltages, separated by a region of uncertainty, rather than by nominal levels alone. The regions of acceptable high and low voltages are delimited by the V_{IH} and V_{IL} voltage levels, respectively. By definition, these represent the points where the gain ($= dV_{out} / dV_{in}$) of the VTC equals -1 as shown in Figure 1-12b. The region between V_{IH} and V_{IL} is called the *undefined region*.

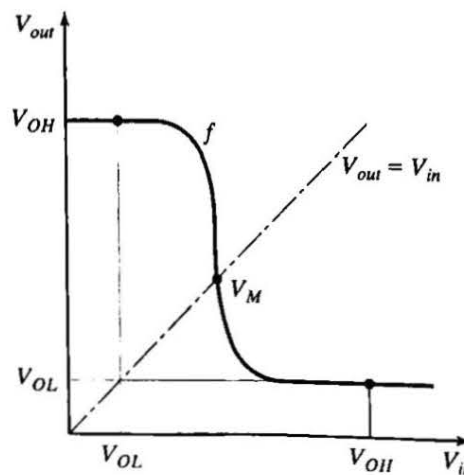
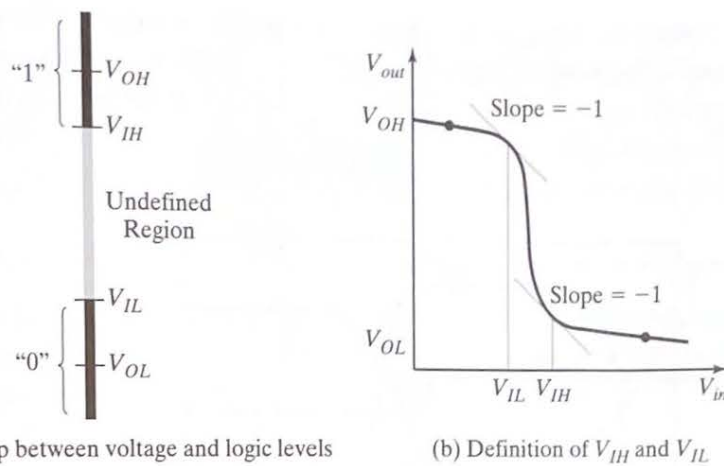


Figure 1-11 Inverter voltage-transfer characteristic.



(a) Relationship between voltage and logic levels

(b) Definition of V_{IH} and V_{IL} **Figure 1-12** Mapping logic levels to the voltage domain.

(sometimes also called the *transition width*, or *TW*). Steady-state signals should avoid this region if proper circuit operation is to be ensured.

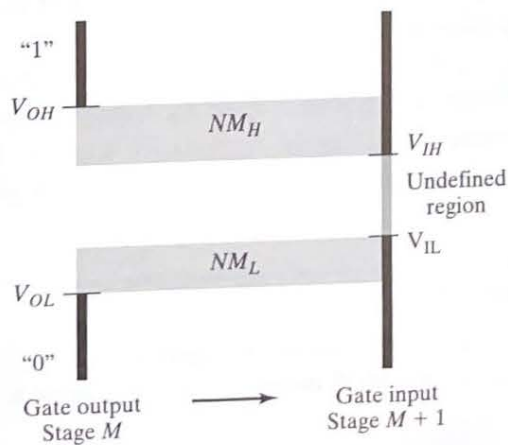
Noise Margins

For a gate to be robust and insensitive to noise disturbances, it is essential that the "0" and "1" intervals be as large as possible. A measure of the sensitivity of a gate to noise is given by the noise margins NM_L (*noise margin low*) and NM_H (*noise margin high*), which quantize the size of the legal "0" and "1," respectively, and set a fixed maximum threshold on the noise value:

$$NM_L = V_{IL} - V_{OL} \quad (1.8)$$

$$NM_H = V_{OH} - V_{IH}$$

The noise margins represent the levels of noise that can be sustained when gates are cascaded as illustrated in Figure 1-13. It is obvious that the margins should be larger than 0 for a digital circuit to be functional, and by preference, they should be as large as possible.

**Figure 1-13** Cascaded inverter gates: definition of noise margins.

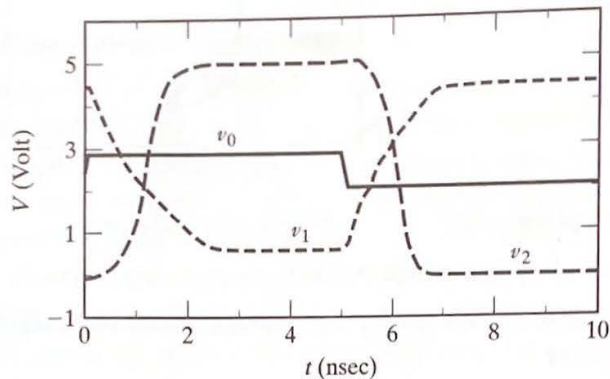
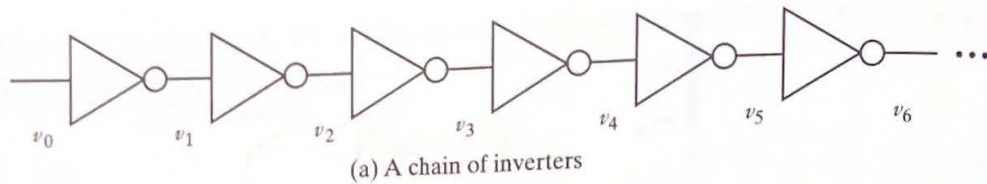


Figure 1-14 The regenerative property.

Regenerative Property

A large noise margin is desirable, but it is not the only requirement. Assume that a signal is disturbed by noise and differs from the nominal voltage levels. As long as the signal is within the noise margins, the gate that follows continues to function correctly, although its output voltage varies from the nominal one. This deviation is added to the noise injected at the output node and passed to the next gate. The effect of different noise sources may accumulate and eventually force a signal level into the undefined region. Fortunately, this does not happen if the gate possesses the *regenerative property*, which ensures that a disturbed signal gradually converges back to one of the nominal voltage levels after passing through a number of logical stages. This property can be understood as follows:

An input voltage v_{in} ($v_{in} \in "0"$) is applied to a chain of N inverters (Figure 1-14a). Assuming that the number of inverters in the chain is even, the output voltage v_{out} ($N \rightarrow \infty$) will equal V_{OL} if and only if the inverter possesses the regenerative property. Similarly, when an input voltage v_{in} ($v_{in} \in "1"$) is applied to the inverter chain, the output voltage will approach the nominal value V_{OH} .

Example 1.4 Regenerative Property

The concept of regeneration is illustrated in Figure 1-14b, which plots the simulated transient response of a chain of CMOS inverters. The input signal to the chain is a step wave-

form with a degraded amplitude, which could be caused by noise. Instead of swinging from rail to rail, v_0 only extends between 2.1 and 2.9 V. From the simulation, it can be observed that this deviation rapidly disappears while progressing through the chain; v_1 , for instance, extends from 0.6 V to 4.45 V. Even further, v_2 already swings between the nominal V_{OL} and V_{OH} . The inverter used in this example clearly possesses the regenerative property.

The conditions under which a gate is regenerative can be intuitively derived by analyzing a simple case study. Figure 1-15a plots the VTC of an inverter $V_{out} = f(V_{in})$ as well as its inverse function $f_{inv}()$, which reverts the function of the x - and y -axis and is defined as

$$in = f(out) \Rightarrow in = f_{inv}(out) \quad (1.9)$$

Assume that a voltage v_0 , deviating from the nominal voltages, is applied to the first inverter in the chain. The output voltage of this inverter equals $v_1 = f(v_0)$ and is applied to the next inverter. Graphically, this corresponds to $v_1 = f_{inv}(v_2)$. The signal voltage gradually converges to the nominal signal after a number of inverter stages, as indicated by the arrows. In Figure 1-15b the signal does not converge to any of the nominal voltage levels, but to an intermediate voltage level. Hence, the characteristic is nonregenerative. The difference between the two cases is due to the gain characteristics of the gates. To be regenerative, the VTC should have a transient region (or undefined region) with a gain *greater than* 1 in absolute value, bordered by the two legal zones, where the gain should be *less than* 1. Such a gate has two stable operating points. This clarifies the definition of the V_{IH} and the V_{IL} levels that form the boundaries between the legal and the transient zones.

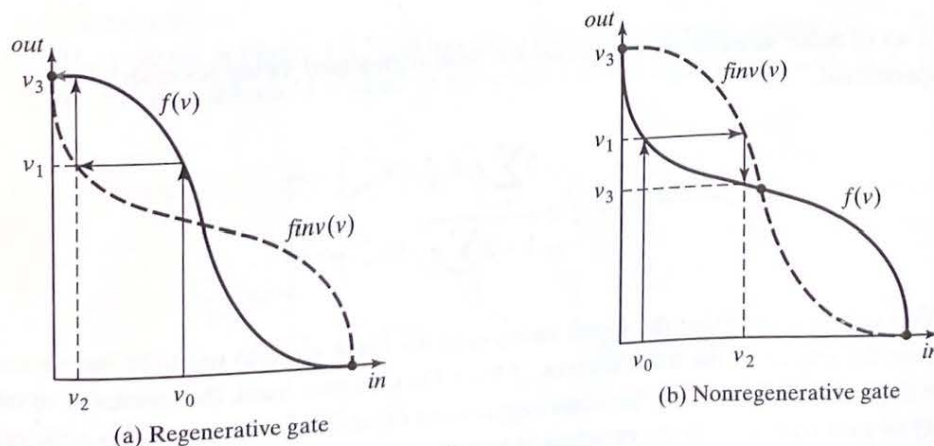


Figure 1-15 Conditions for regeneration.

Noise Immunity

While the noise margin is a meaningful means for measuring the robustness of a circuit against noise, it is not sufficient. It expresses the capability of a circuit to “overpower” a noise source. *Noise immunity*, on the other hand, expresses the ability of the system to process and transmit information correctly in the presence of noise [Dally98]. Many digital circuits with low noise margins have very good noise immunity because they *reject a noise source* rather than overpower it. These circuits have the property that only a small fraction of a potentially damaging noise source is coupled to the important circuit nodes. More precisely, the transfer function between noise source and signal node is far less than 1. Circuits that do not possess this property are *susceptible* to noise.

To study the noise immunity of a gate, we have to construct a noise budget that allocates the power budget to the various noise sources. As discussed earlier, the noise sources can be divided into the following types of sources:

- those that are *proportional* to the signal swing V_{sw} . The impact on the signal node is expressed as $g V_{sw}$; and
- those that are *fixed*. The impact on the signal node equals $f V_{Nfi}$, with V_{Nfi} the amplitude of the noise source, and f the transfer function *from noise* to signal node.

We assume, for the sake of simplicity, that the noise margin equals half the signal swing (for both H and L). To operate correctly, the noise margin has to be larger than the sum of the coupled noise values:

$$V_{NM} = \frac{V_{sw}}{2} \geq \sum_i f_i V_{Nfi} + \sum_j g_j V_{sw} \quad (1.10)$$

Given a set of noise sources, we can derive the minimum signal swing necessary for the system to be operational:

$$V_{sw} \geq \frac{2 \sum_i f_i V_{Nfi}}{1 - 2 \sum_j g_j} \quad (1.11)$$

This makes it clear that the signal swing (and the noise margin) has to be large enough to overpower the impact of the fixed sources ($f V_{Nfi}$). On the other hand, the sensitivity to internal sources depends primarily upon the noise suppressing capabilities of the gate, this is the proportionality or gain factors g_j . In the presence of large gain factors, increasing the signal swing does not do any good to suppress noise, as the noise increases proportionally. In later chapters, we will discuss some differential logic families that suppress most of the internal noise, and thus can get away with very small noise margins and signal swings.

Directivity

The directivity property requires a gate to be *unidirectional*—that is, changes in an output level should not appear at any unchanging input of the same circuit. Otherwise, an output-signal transition reflects to the gate inputs as a noise signal, affecting the signal integrity.

In real gate implementations, full directivity can never be achieved. Some feedback of changes in output levels to the inputs cannot be avoided. Capacitive coupling between inputs and outputs is a typical example of such a feedback. It is important to minimize these changes so that they do not affect the logic levels of the input signals.

Fan-In and Fan-Out

The *fan-out* denotes the number of load gates N that are connected to the output of the driving gate (see Figure 1-16). Increasing the fan-out of a gate can affect its logic output levels. From the world of analog amplifiers, we know that this effect is minimized by making the input resistance of the load gates as large as possible (minimizing the input currents) and by keeping the output resistance of the driving gate small (reducing the effects of load currents on the output voltage). When the fan-out is large, the added load can deteriorate the dynamic performance of the driving gate. For these reasons, many generic and library components define a *maximum fan-out* to guarantee that the static and dynamic performance of the element meet specification.

The *fan-in* of a gate is defined as the number of inputs to the gate (see Figure 1-16b). Gates with large fan-in tend to be more complex, which often results in inferior static and dynamic properties.

The Ideal Digital Gate

Based on these observations, we can define the *ideal* digital gate from a static perspective. The ideal inverter model is important because it gives us a metric by which we can judge the quality of actual implementations.

Its VTC is shown in Figure 1-17 and has the following properties: infinite gain in the transition region, and gate threshold located in the middle of the logic swing, with high and low

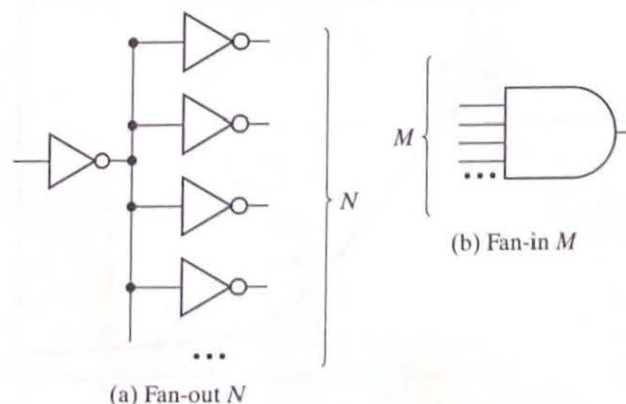


Figure 1-16 Definition of fan-out and fan-in of a digital gate.

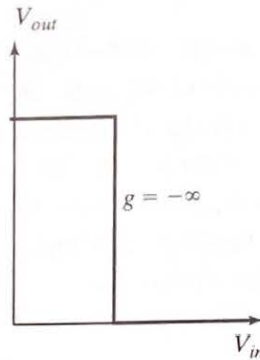


Figure 1-17 Ideal voltage-transfer characteristic.

noise margins equal to half the swing. The input and output impedances of the ideal gate are infinity and zero, respectively (i.e., the gate has unlimited fan-out). While this ideal VTC is unfortunately impossible in real designs, some implementations, such as the static CMOS inverter, come close.

Example 1.5 Voltage-Transfer Characteristic

Figure 1-18 shows an example of a voltage-transfer characteristic of an actual, but outdated, gate structure. The values of the dc parameters are derived from inspection of the graph.

$$\begin{aligned} V_{OH} &= 3.5 \text{ V}; & V_{OL} &= 0.45 \text{ V} \\ V_{IH} &= 2.35 \text{ V}; & V_{IL} &= 0.66 \text{ V} \\ V_M &= 1.64 \text{ V} \\ NM_H &= 1.15 \text{ V}; & NM_L &= 0.21 \text{ V} \end{aligned}$$

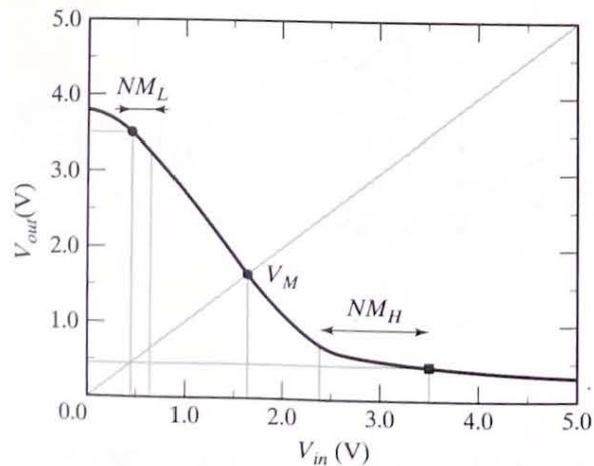


Figure 1-18 Voltage-transfer characteristic of an NMOS inverter of the 1970s.

The observed transfer characteristic, obviously, is far from ideal: it is asymmetrical, has a very low value for NM_L , and the voltage swing of 3.05 V is substantially below the maximum obtainable value of 5 V (which is the value of the supply voltage for this design).

1.3.3 Performance

From a system designer's perspective, the performance of a digital circuit expresses its computational ability. For instance, a microprocessor often is characterized by the number of instructions it can execute per second. This performance metric depends both on the architecture of the processor—for instance, the number of instructions it can execute in parallel—and the actual design of logic circuitry. While the former is crucially important, it is not the focus of this text. The reader may refer to the many excellent books on this topic [for instance, Hennessy02]. When focusing on the pure design, performance is most often expressed by the duration of the clock period (*clock cycle time*), or its rate (*clock frequency*). The minimum value of the clock period for a given technology and design is set by a number of factors such as the time it takes for the signals to propagate through the logic, the time it takes to get the data in and out of the registers, and the uncertainty of the clock arrival times. Each of these topics will be discussed in detail in this book. The performance of an individual gate lies at the core of the whole performance analysis, however.

The *propagation delay* t_p of a gate defines how quickly it responds to a change at its input(s).^{*} It expresses the delay experienced by a signal when passing through a gate. It is measured between the 50% transition points of the input and output waveforms, as shown in Figure 1-19 for

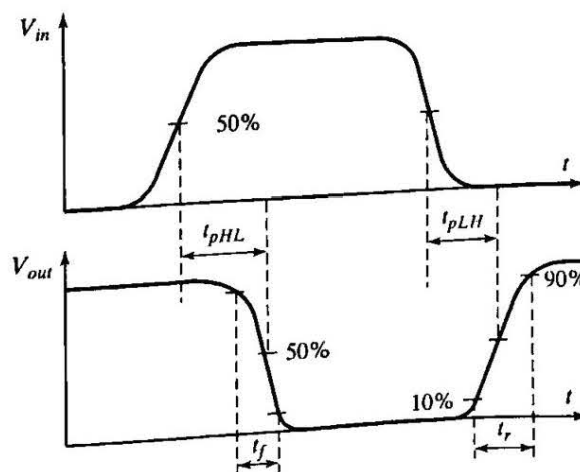


Figure 1-19 Definition of propagation delays and rise and fall times.

an inverting gate.² Because a gate displays different response times for rising or falling input waveforms, two definitions of the propagation delay are necessary. The t_{pLH} defines the response time of the gate for a *low-to-high* (or positive) output transition, while t_{pHL} refers to a *high-to-low* (or negative) transition. The propagation delay t_p is defined as the average of the two:

$$t_p = \frac{t_{pLH} + t_{pHL}}{2} \quad (1.12)$$

CAUTION: Observe that the propagation delay t_p , in contrast to t_{pLH} and t_{pHL} , is an artificial gate quality metric, and has no physical meaning per se. It is mostly used to compare different semiconductor technologies, circuit, or logic design styles.

The propagation delay is not only a function of the circuit technology and topology, but depends upon other factors as well. Most importantly, the delay is a function of the *slopes* of the input and output signals of the gate. To quantify these properties, we introduce the *rise and fall times*, t_r and t_f , which are metrics that apply to individual signal waveforms rather than to gates (see Figure 1-19), and they express how fast a signal transits between the different levels. The uncertainty over when a transition actually starts or ends is avoided by defining the rise and fall times between the 10% and 90% points of the waveforms, as shown in the figure. The rise/fall time of a signal is largely determined by the strength of the driving gate, and the load presented to it.

When comparing the performance of gates implemented in different technologies or circuit styles, it is important not to confuse the picture by including parameters such as load factors, fan-in, and fan-out. A uniform way of measuring the t_p of a gate so that technologies can be judged on an equal footing is desirable. The de facto standard circuit for delay measurement is the *ring oscillator*, which consists of an odd number of inverters connected in a circular chain (see Figure 1-20.) Due to the odd number of inversions, this circuit does not have a stable operating point and oscillates. The period T of the oscillation is determined by the propagation time of a signal transition through the complete chain, or $T = 2 \times t_p \times N$ with N the number of inverters in the chain. The factor 2 results from the observation that a full cycle requires both a low-to-high and a high-to-low transition. Note that this equation is only valid for $2Nt_p \gg t_f + t_r$. If this condition is not met, the circuit might not oscillate—one “wave” of signals propagating through the ring will overlap with a successor and eventually dampen the oscillation. Typically, a ring oscillator needs at least five stages to be operational.

²The 50% definition is inspired by the assumption that the switching threshold V_M is typically located in the middle of the logic swing.

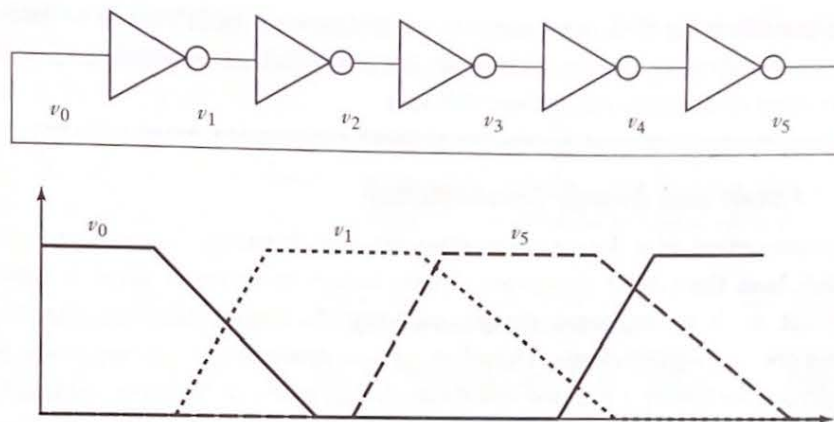


Figure 1-20 Ring oscillator circuit for propagation-delay measurement.

CAUTION: We must be extremely careful with results obtained from ring oscillator measurements. A t_p of 20 ps by no means implies that a circuit built with those gates will operate at 50 GHz. The oscillator results are primarily useful for quantifying the differences between various manufacturing technologies and gate topologies. The oscillator is an idealized circuit in which each gate has a fan-in and fan-out of exactly 1 and parasitic loads are minimal. In more realistic digital circuits, fan-ins and fan-outs are higher, and interconnect delays are non-negligible. The gate functionality is also substantially more complex than a simple invert operation. As a result, the achievable clock frequency, on average, is 50 to 100 times slower than the frequency obtained from ring oscillator measurements. This is an average observation; carefully optimized designs might approach use ideal frequency more closely.

Example 1.6 Propagation Delay of First-Order RC Network

Digital circuits are often modeled as first-order RC networks of the type shown in Figure 1-21. The propagation delay of such a network is thus of considerable interest.

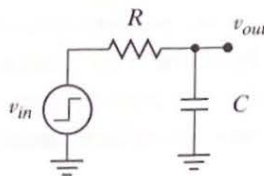


Figure 1-21 First-order RC network.

When applying a step input (with v_{in} going from 0 to V), the transient response of this circuit is known to be an exponential function, and is given by the following expression (where $\tau = RC$, the time constant of the network):

$$v_{out}(t) = (1 - e^{-t/\tau}) V \quad (1.13)$$

The time to reach the 50% point is easily computed as $t = \ln(2)\tau = 0.69\tau$. Similarly, it takes $t = \ln(9)\tau = 2.2\tau$ to get to the 90% point. It is worth memorizing these numbers because they are used extensively throughout this text.

1.3.4 Power and Energy Consumption

The power consumption of a design determines how much energy is consumed per operation, and how much heat the circuit dissipates. These factors influence a great number of critical design decisions, such as the power supply capacity, the battery lifetime, supply line sizing, packaging and cooling requirements. Therefore, power dissipation is an important property of a design that affects feasibility, cost, and reliability. In the world of high-performance computing, power consumption limits, dictated by the chip package and the heat removal system, determine the number of circuits that can be integrated onto a single chip, and how fast they are allowed to switch. With the increasing popularity of mobile and distributed computation, energy limitations put a firm restriction on the number of computations that can be performed given a minimum time between battery recharges.

Depending on the design problem at hand, different dissipation measures must be considered. For instance, the peak power P_{peak} is important when studying supply line sizing. When addressing cooling or battery requirements, one is predominantly interested in the average power dissipation P_{av} . The measures are defined as

$$\begin{aligned} P_{peak} &= i_{peak} V_{supply} = \max[p(t)] \\ P_{av} &= \frac{1}{T} \int_0^T p(t) dt = \frac{V_{supply}}{T} \int_0^T i_{supply}(t) dt \end{aligned} \quad (1.14)$$

where $p(t)$ is the instantaneous power, i_{supply} is the current being drawn from the supply voltage V_{supply} over the interval $t \in [0, T]$, and i_{peak} is the maximum value of i_{supply} over that interval.

The dissipation can further be decomposed into *static* and *dynamic* components. The latter occurs only during transients, when the gate is switching. It is attributed to the charging of capacitors and temporary current paths between the supply rails; therefore it is proportional to the switching frequency: *the higher the number of switching events, the greater the dynamic power consumption*. On the other hand, the static component is present even when no switching occurs and is caused by static conductive paths between the supply rails or by leakage currents. It is always present, even when the circuit is in standby. Minimization of this consumption source is a worthwhile goal.

The propagation delay and the power consumption of a gate are related—the propagation delay is mostly determined by the speed at which a given amount of energy can be stored on the gate capacitors. The faster the energy transfer (or the higher the power consumption), the faster the gate. For a given technology and gate topology, the product of power consumption and propagation delay is generally a constant. This product is called the *power-delay product* (or PDP),

and can be considered as a quality measure for a switching device. The PDP is simply the *energy consumed by the gate per switching event*. The ring oscillator is again the circuit of choice for measuring the PDP of a logic family.

An ideal gate is one that is fast and consumes little energy. The *energy-delay product (E-D)* is a combined metric that brings those two elements together, and is often used as the ultimate quality metric. Thus, it should be clear that the E-D is equivalent to *power-delay*².

Example 1.7 Energy Dissipation of First-Order RC Network

Let us again consider the first-order RC network (shown in Figure 1-21). When applying a step input (with V_{in} going from 0 to V), an amount of energy is provided by the signal source to the network. The total energy delivered by the source (from the start of the transition to the end) can be readily computed:

$$E_{in} = \int_0^{\infty} i_{in}(t)v_{in}(t)dt = V \int_0^{\infty} C \frac{dv_{out}}{dt} dt = (CV) \int_0^V dv_{out} = CV^2 \quad (1.15)$$

It is interesting to observe that the energy needed to charge a capacitor from 0 to V volts with a step input is a function of the size of the voltage step and the capacitance, but is independent of the value of the resistor. We can also compute how much of the delivered energy gets stored on the capacitor at the end of the transition.

$$E_C = \int_0^{\infty} i_C(t)v_{out}(t)dt = \int_0^{\infty} C \frac{dv_{out}}{dt} v_{out} dt = C \int_0^V v_{out} dv_{out} = \frac{CV^2}{2} \quad (1.16)$$

This is exactly half of the energy delivered by the source. For those who wonder what happened to the other half—a simple analysis shows that an equivalent amount gets dissipated as heat in the resistor during the transaction. It is left to the reader to demonstrate that during the discharge phase (for a step from V to 0), the energy originally stored on the capacitor gets dissipated in the resistor as well, and then turned into heat.

1.4 Summary

In this introductory chapter, we learned about the history and the trends in digital circuit design. We also introduced the important quality metrics used to evaluate the quality of a design: cost, functionality, robustness, performance, and energy/power dissipation.

1.5 To Probe Further

The design of digital integrated circuits has been the topic of many textbooks and monographs. To help the reader find more information on some selected topics, an extensive list of references

follows. The state-of-the-art developments in the area of digital design are generally reported in technical journals or conference proceedings, the most important of which are listed.

Journals and Proceedings

IEEE Journal of Solid-State Circuits

IEICE Transactions on Electronics (Japan)

Proceedings of The International Solid-State and Circuits Conference (ISSCC)

Proceedings of the VLSI Circuits Symposium

Proceedings of the Custom Integrated Circuits Conference (CICC)

European Solid-State Circuits Conference (ESSCIRC)

Bibliography

MOS

M. Annaratone, *Digital CMOS Circuit Design*, Kluwer, 1986.

T. Dillinger, *VLSI Engineering*, Prentice Hall, 1988.

M. Elmasry, ed., *Digital MOS Integrated Circuits*, IEEE Press, 1981.

M. Elmasry, ed., *Digital MOS Integrated Circuits II*, IEEE Press, 1992.

L. Glasser and D. Doppelpuhl, *The Design and Analysis of VLSI Circuits*, Addison-Wesley, 1985.

A. Kang and Leblebici, *CMOS Digital Integrated Circuits*, 2nd Ed., McGraw-Hill, 1999.

C. Mead and L. Conway, *Introduction to VLSI Systems*, Addison-Wesley, 1980.

K. Martin, *Digital Integrated Circuit Design*, Oxford University Press, 2000.

D. Pucknell and K. Eshraghian, *Basic VLSI Design*, Prentice Hall, 1988.

M. Shoji, *CMOS Digital Circuit Technology*, Prentice Hall, 1988.

J. Uyemura, *Circuit Design for CMOS VLSI*, Kluwer, 1992.

H. Veendrick, *Deep-Submicron CMOS IC's: From Basics to ASICs*, Second Edition, Kluwer Academic Publishers, 2000.

N. Weste and K. Eshraghian, *Principles of CMOS VLSI Design*, Addison-Wesley, 1985, 1993.

High-Performance Design

K. Bernstein et al, *High Speed CMOS Design Styles*, Kluwer Academic, 1998.

A. Chandrakasan, F. Fox, and W. Bowhill, ed., *Design of High-Performance Microprocessor Circuits*, IEEE Press, 2000.

M. Shoji, *High-Speed Digital Circuits*, Addison-Wesley, 1996.

Low-Power Design

A. Chandrakasan and R. Brodersen, ed., *Low-Power Digital CMOS Design*, IEEE Press, 1998.

M. Pedram and J. Rabaey, ed., *Power-Aware Design Methodologies*, Kluwer Academic, 2002.

J. Rabaey and M. Pedram, ed., *Low-Power Design Methodologies*, Kluwer Academic, 1996.

G. Yeap, *Practical Low-Power CMOS Design*, Kluwer Academic, 1998.

Memory Design

K. Itoh, *VLSI Memory Chip Design*, Springer, 2001.

B. Keeth and R. Baker, *DRAM Circuit Design*, IEEE Press, 1999.

B. Prince, *Semiconductor Memories*, Wiley, 1991.

B. Prince, *High Performance Memories*, Wiley, 1996.

D. Hodges, *Semiconductor Memories*, IEEE Press, 1972.

Interconnections and Packaging

- H. Bakoglu, *Circuits, Interconnections, and Packaging for VLSI*, Addison-Wesley, 1990.
 W. Dally and J. Poulton, *Digital Systems Engineering*, Cambridge University Press, 1998.
 E. Friedman, ed., *Clock Distribution Networks in VLSI Circuits and Systems*, IEEE Press, 1995.
 J. Lau et al, ed., *Electronic Packaging: Design, Materials, Process, and Reliability*, McGraw-Hill, 1998.

Design Tools and Methodologies

- V. Agrawal and S. Seth, *Test Generation for VLSI Chips*, IEEE Press, 1988.
 D. Clein, *CMOS IC Layout*, Newnes, 2000.
 G. De Micheli, *Synthesis and Optimization of Digital Circuits*, McGraw-Hill, 1994.
 S. Rubin, *Computer Aids for VLSI Design*, Addison-Wesley, 1987.
 J. Uyemura, *Physical Design of CMOS Integrated Circuits Using L-Edit*, PWS, 1995.
 A. Vladimirescu, *The Spice Book*, John Wiley and Sons, 1993.
 W. Wolf, *Modern VLSI Design*, Prentice Hall, 1998.

Bipolar and BiCMOS

- A. Alvarez, *BiCMOS Technology and Its Applications*, Kluwer, 1989.
 M. Elmasry, ed., *BiCMOS Integrated Circuit Design*, IEEE Press, 1994.
 S. Embabi, A. Bellaouar, and M. Elmasry, *Digital BiCMOS Integrated Circuit Design*, Kluwer, 1993.

General

- J. Buchanan, *CMOS/TTL Digital Systems Design*, McGraw-Hill, 1990.
 H. Haznedar, *Digital Micro-Electronics*, Benjamin/Cummings, 1991.
 D. Hodges and H. Jackson, *Analysis and Design of Digital Integrated Circuits*, 2nd ed., McGraw-Hill, 1988.
 M. Smith, *Application-Specific Integrated Circuits*, Addison-Wesley, 1997.
 R. K. Watts, *Submicron Integrated Circuits*, Wiley, 1989.

References

- [Bardeen48] J. Bardeen and W. Brattain, "The Transistor, a Semiconductor Triode," *Phys. Rev.*, vol. 74, p. 230, July 15, 1948.
 [Beeson62] R. Beeson and H. Ruegg, "New Forms of All Transistor Logic," *ISSCC Digest of Technical Papers*, pp. 10-11, Feb. 1962.
 [Dally98] B. Dally, *Digital Systems Engineering*, Cambridge University Press, 1998.
 [Faggin72] F. Faggin, M.E. Hoff, Jr, H. Feeney, S. Mazor, M. Shima, "The MCS-4 - An LSI MicroComputer System," 1972 IEEE Region Six Conference Record, San Diego, CA, pp. 1-6, April 1972.
 [Harris56] J. Harris, "Direct-Coupled Transistor Logic Circuitry in Digital Computers," *ISSCC Digest of Technical Papers*, p. 9, Feb. 1956.
 [Hart72] C. Hart and M. Slob, "Integrated Injection Logic—A New Approach to LSI," *ISSCC Digest of Technical Papers*, pp. 92-93, Feb. 1972.
 [Hoff70] E. Hoff, "Silicon-Gate Dynamic MOS Crams 1,024 Bits on a Chip," *Electronics*, pp. 68-73, August 3, 1970.
 [Intel01] "Moore's Law", <http://www.intel.com/research/silicon/mooreslaw.htm>
 [Masaki74] A. Masaki, Y. Harada and T. Chiba, "200-Gate ECL Master-Slice LSI," *ISSCC Digest of Technical Papers*, pp. 62-63, Feb. 1974.
 [Moore65] G. Moore, "Cramming more Components into Integrated Circuits," *Electronics*, Vol. 38, Nr 8, April 1965.
 [Murphy93] B. Murphy, "Perspectives on Logic and Microprocessors," *Commemorative Supplement to the Digest of Technical Papers, ISSCC*, pp. 49-51, San Francisco, 1993.

- [Norman60] R. Norman, J. Last and I. Haas, "Solid-State Micrologic Elements," *ISSCC Digest of Technical Papers*, pp. 82–83, Feb. 1960.
- [Hennessy02] J. Hennessy, D. Patterson, and David Goldberg, *Computer Architecture A Quantitative Approach*, Third Edition, Morgan Kaufmann Publishers, 2002.
- [Saleh01] R. Saleh, M. Benoit, and P. McCrorie, "Power Distribution Planning", Simplex Solutions, http://www.simplex.com/wt/sec.php?page_name=wp_powerplan
- [Schockley49] W. Schockley, "The Theory of pn Junctions in Semiconductors and pn-Junction Transistors," *BSTJ*, vol. 28, p. 435, 1949.
- [Shima74] M. Shima, F. Faggin and S. Mazor, "An N-Channel, 8-bit Single-Chip Microprocessor," *ISSCC Digest of Technical Papers*, pp. 56–57, Feb. 1974.
- [Swade93] D. Swade, "Redeeming Charles Babbage's Mechanical Computer," *Scientific American*, pp. 86–91, February 1993.
- [Wanlass63] F. Wanlass, and C. Sah, "Nanowatt logic Using Field-Effect Metal-Oxide Semiconductor Triodes," *ISSCC Digest of Technical Papers*, pp. 32–32, Feb. 1963.

Exercises

Please refer to <http://bwrc.eecs.berkeley.edu/IcBook> for up-to-date problem sets and exercises. By making the exercises electronically available rather than in print, we can provide a dynamic environment that tracks the rapid evolution of today's digital integrated circuit design technology.