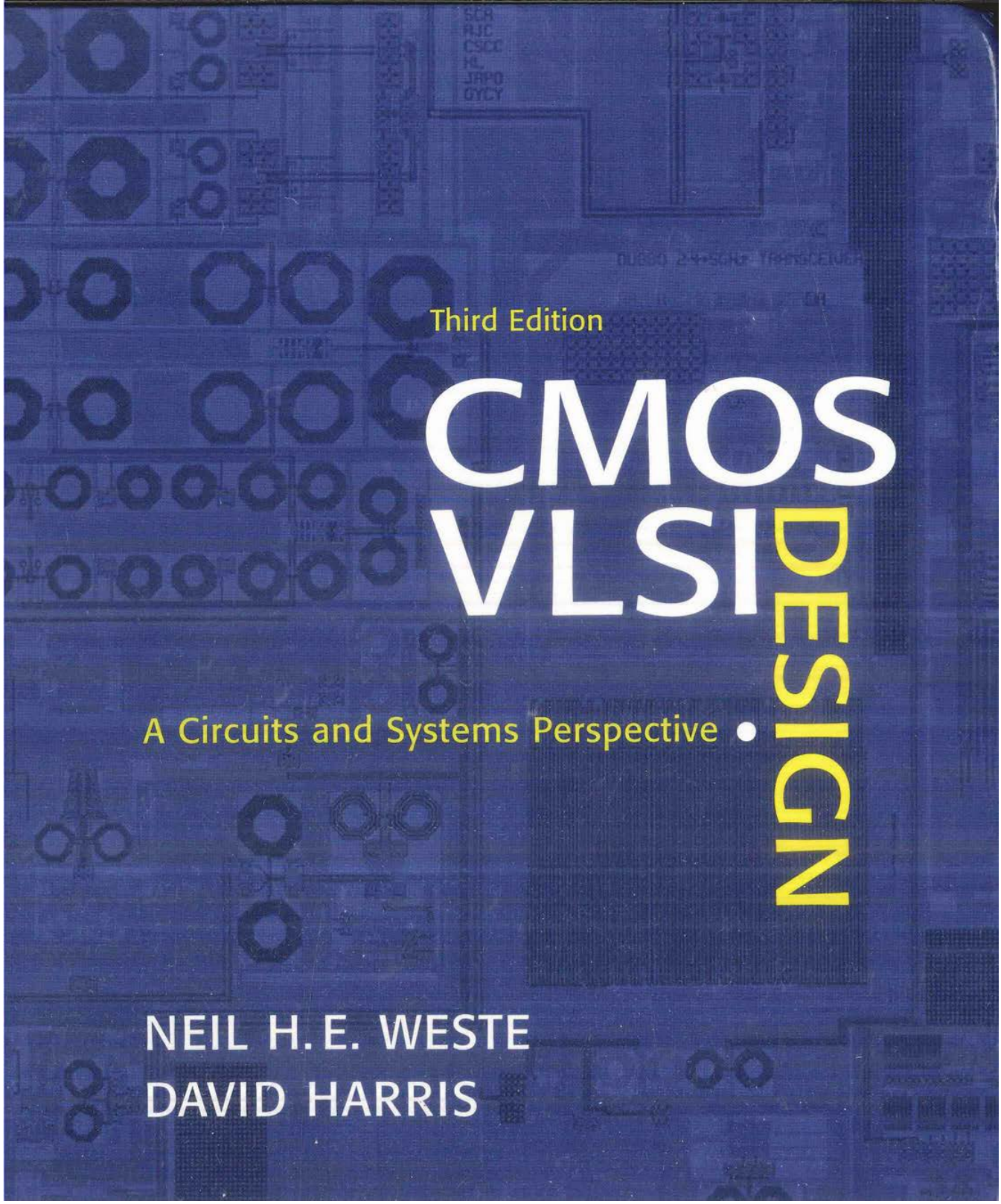


TSMC-1041
U.S. Patent 7,923,764

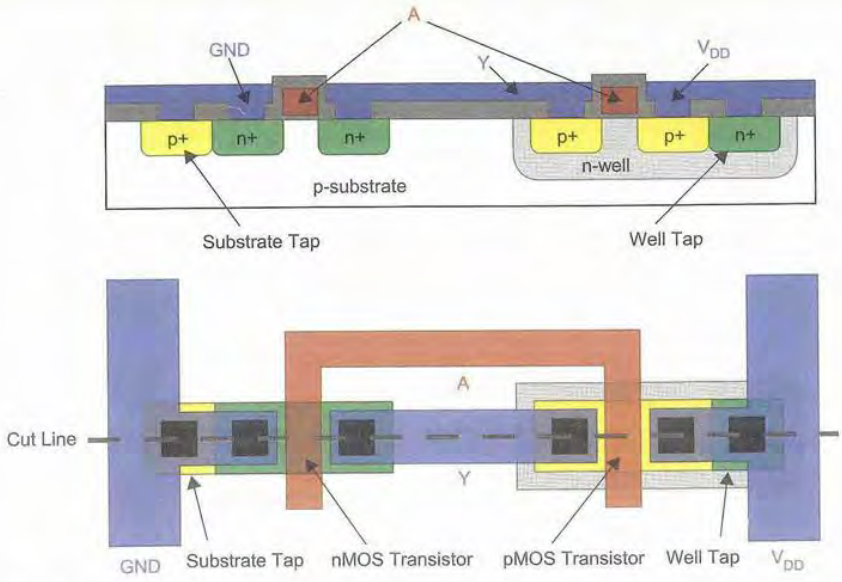
CMOS VLSI DESIGN

A Circuits and Systems Perspective • Third Edition

Weste/Harris



Addison



Figs 1.34–1.35(a) Inverter Cross-Section and Top View

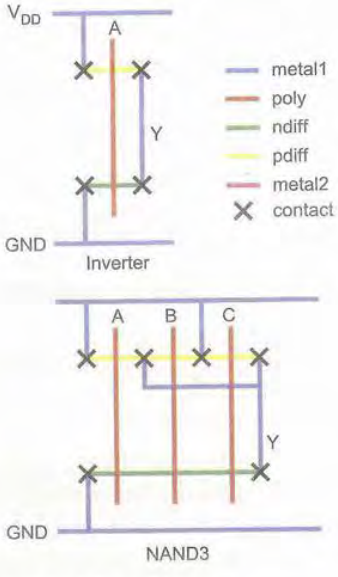


Fig 1.43 Stick Diagrams

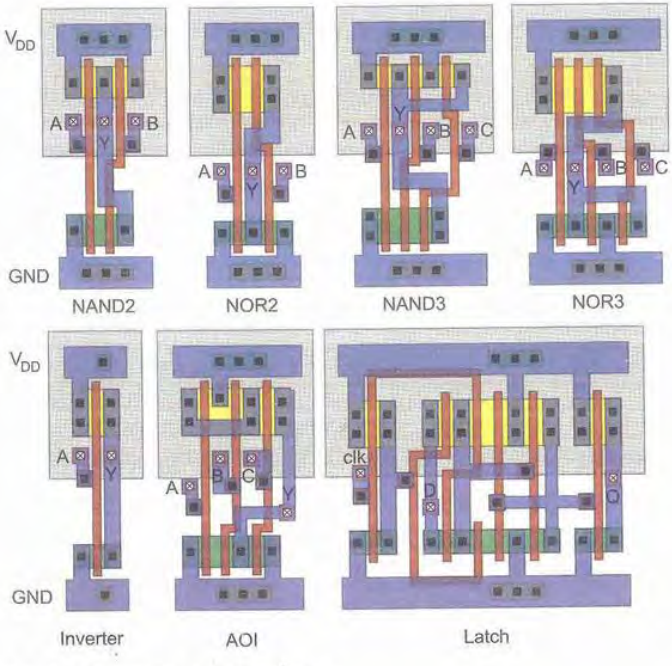


Fig 1.62 Standard Cell Layouts

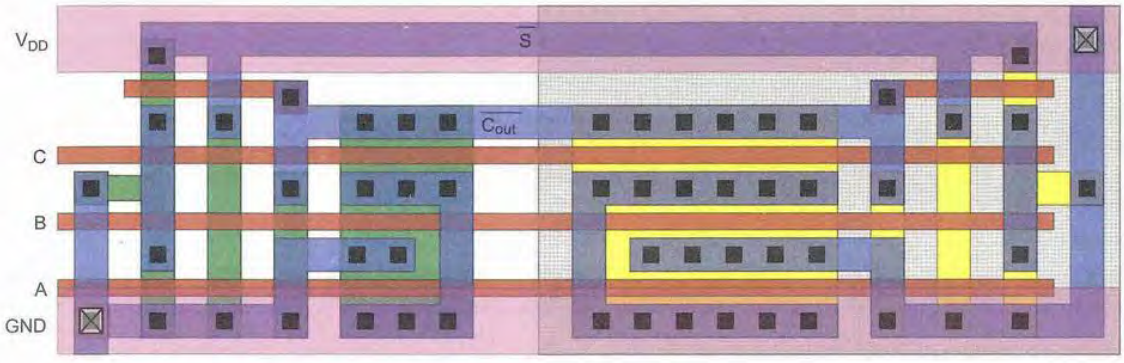


Fig 10.5(b) Full Adder Layout

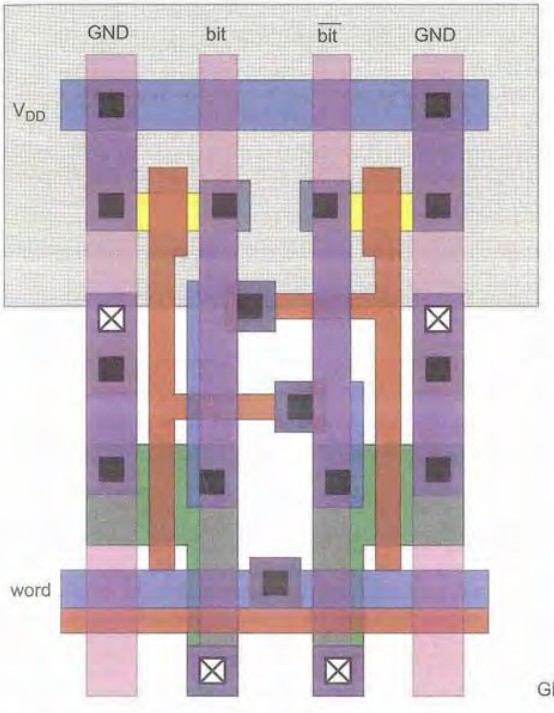


Fig 11.6(a) 6T SRAM Layout

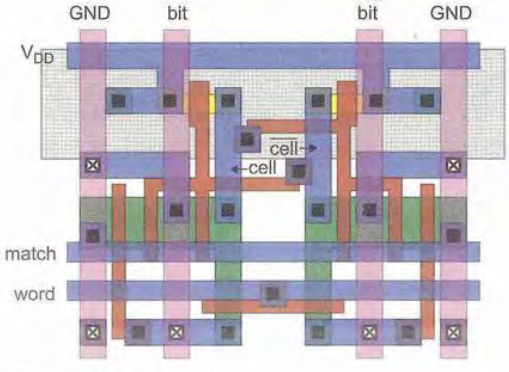


Fig 11.50 CAM Layout

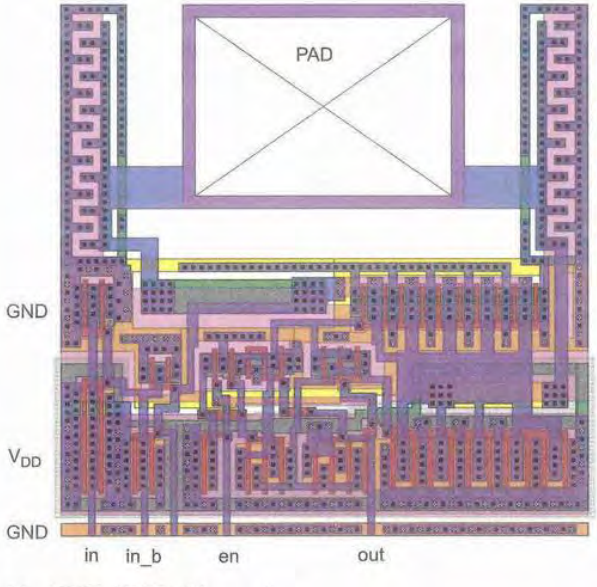


Fig 12.23 I/O Pad Layout

CMOS VLSI Design

A Circuits and Systems Perspective

Third Edition

Neil H. E. Weste
Macquarie University and
The University of Adelaide

David Harris
Harvey Mudd College



Boston San Francisco New York
London Toronto Sydney Tokyo Singapore Madrid
Mexico City Munich Paris Cape Town Hong Kong Montreal

Sponsoring Editor: Matt Goldstein
Project Editor: Maite Suarez-Rivas
Senior Production Supervisor: Juliet Silveri
Production Services and Copyediting: Kathy Smith
Proofreading: Holly McLean-Aldis
Composition and Art: Gillian Hall, The Aardvark Group
Cover Design Supervisor: Joyce Cosentino Wells
Cover Designer: Alison R. Paddock
Text Designer: Delgado and Company, Inc.
Marketing Manager: Michelle Brown
Print Buyer: Caroline Fell

The photograph used on the cover and chapter openers is copyright ©2004 by Cisco Systems Inc., Wireless Networking Business Unit. Used with permission.

Access the latest information about Addison-Wesley titles from our World Wide Web site: http://www.aw-bc.com/computing

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and Addison-Wesley was aware of a trademark claim, the designations have been printed in initial caps or all caps.

The programs and applications presented in this book have been included for their instructional value. They have been tested with care, but are not guaranteed for any particular purpose. The publisher does not offer any warranties or representations, nor does it accept any liabilities with respect to the programs or applications.

Library of Congress Cataloging-in-Publication Data
Weste, Neil H. E.
CMOS VLSI design : a circuits and systems perspective / Neil H.E. Weste, David Harris.-- 3rd ed.
p. cm.
Includes bibliographical references and index.
ISBN 0-321-14901-7
1. Rev. ed. of: Principles of CMOS VLSI design. 1993. 2. Integrated circuits--Very large scale integration--design and construction. 3. Metal oxide semiconductors, Complementary. I. Harris, David (David F.) II. Weste, Neil H. E. Principles of CMOS VLSI design. III. Title.

TK7874.W45 2005
621.39'5--dc22

2004007092

For information on obtaining permission for the use of material from this work, please submit a written request to Pearson Education, Inc., Rights and Contracts Department, 75 Arlington St., Suite 300, Boston, MA 02116 or fax your request to 617-848-7047.

Copyright © 2005 by Pearson Education, Inc.

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the publisher. Printed in the United States of America.

1 2 3 4 5 6 7 8 9 10—CRW—060504

To the littlest princesses
Jocelyn and Makayla (N.W.)

To Jennifer (D.H.)

Contents

CHAPTER 1 Introduction

1.1 A Brief History 1
1.2 Book Summary 5
1.3 MOS Transistors 7
1.4 CMOS Logic 10
- **1.4.1** The Inverter 10
- **1.4.2** The NAND Gate 10
- **1.4.3** Combinational Logic 11
- **1.4.4** The NOR Gate 12
- **1.4.5** Compound Gates 13
- **1.4.6** Pass Transistors and Transmission Gates 14
- **1.4.7** Tristates 17
- **1.4.8** Multiplexers 18
- **1.4.9** Latches and Flip-Flops 20

1.5 CMOS Fabrication and Layout 23
- **1.5.1** Inverter Cross-section 23
- **1.5.2** Fabrication Process 24
- **1.5.3** Layout Design Rules 28
- **1.5.4** Gate Layout 32
- **1.5.5** Stick Diagrams 33

1.6 Design Partitioning 35
1.7 Example: A Simple MIPS Microprocessor 39
- **1.7.1** MIPS Architecture 39
- **1.7.2** Multicycle MIPS Microarchitecture 42

1.8 Logic Design 46
- **1.8.1** Top-level Interface 46
- **1.8.2** Block Diagram 47
- **1.8.3** Hierarchy 47
- **1.8.4** Hardware Description Languages 48

1.9 Circuit Design 49
1.10 Physical Design 52

1.10.1 Floorplanning 52
1.10.2 Standard Cells 55
1.10.3 Snap-together Cells 55
1.10.4 Slice Plans 59
1.10.5 Area Estimation 59
1.11 Design Verification 60
1.12 Fabrication, Packaging, and Testing 61
Summary 63
Exercises 63

CHAPTER 2 MOS Transistor Theory

2.1 Introduction 67
2.2 Ideal I-V Characteristics 71
2.3 C-V Characteristics 75
2.3.1 Simple MOS Capacitance Models 75
2.3.2 Detailed MOS Gate Capacitance Model 77
2.3.3 Detailed MOS Diffusion Capacitance Model 80
2.4 Nonideal I-V Effects 83
2.4.1 Velocity Saturation and Mobility Degradation 84
2.4.2 Channel Length Modulation 86
2.4.3 Body Effect 87
2.4.4 Subthreshold Conduction 88
2.4.5 Junction Leakage 89
2.4.6 Tunneling 90
2.4.7 Temperature Dependence 90
2.4.8 Geometry Dependence 92
2.4.9 Summary 92
2.5 DC Transfer Characteristics 94
2.5.1 Complementary CMOS Inverter DC Characteristics 94
2.5.2 Beta Ratio Effects 97
2.5.3 Noise Margin 98
2.5.4 Ratioed Inverter Transfer Function 100
2.5.5 Pass Transistor DC Characteristics 101
2.5.6 Tristate Inverter 102
2.6 Switch-level RC Delay Models 103
2.7 Pitfalls and Fallacies 106
Summary 107
Exercises 108

CHAPTER 3 CMOS Processing Technology

3.1 Introduction 113
3.2 CMOS Technologies 113
3.2.1 Background 113
3.2.2 Wafer Formation 114
3.2.3 Photolithography 115
3.2.4 Well and Channel Formation 117
3.2.5 Silicon Dioxide (SiO2) 118
3.2.6 Isolation 119
3.2.7 Gate Oxide 120
3.2.8 Gate and Source/Drain Formation 121
3.2.9 Contacts and Metallization 124
3.2.10 Passivation 124
3.2.11 Metrology 125
3.3 Layout Design Rules 125
3.3.1 Design Rule Background 126
3.3.2 Scribe Line and Other Structures 130
3.3.3 MOSIS Scalable CMOS Design Rules 130
3.3.4 Micron Design Rules 134
3.4 CMOS Process Enhancements 136
3.4.1 Transistors 136
3.4.2 Interconnect 140
3.4.3 Circuit Elements 141
3.4.4 Beyond Conventional CMOS 148
3.5 Technology-related CAD Issues 148
3.5.1 Design Rule Checking (DRC) 149
3.5.2 Circuit Extraction 150
3.6 Manufacturing Issues 151
3.6.1 Antenna Rules 151
3.6.2 Layer Density Rules 152
3.6.3 Resolution Enhancement Rules 153
3.7 Pitfalls and Fallacies 153
3.8 Historical Perspective 154
Summary 154
Exercises 154

CHAPTER 4 Circuit Characterization and Performance Estimation

4.1 Introduction 157

4.2 Delay Estimation 158

- **4.2.1** RC Delay Models 159
- **4.2.2** Linear Delay Model 165
- **4.2.3** Logical Effort 166
- **4.2.4** Parasitic Delay 167
- **4.2.5** Limitations to the Linear Delay Model 169

4.3 Logical Effort and Transistor Sizing 173

- **4.3.1** Delay in a Logic Gate 173
- **4.3.2** Delay in Multistage Logic Networks 174
- **4.3.3** Choosing the Best Number of Stages 178
- **4.3.4** Example 181
- **4.3.5** Summary and Observations 183
- **4.3.6** Limitations of Logical Effort 185
- **4.3.7** Extracting Logical Effort from Datasheets 185

4.4 Power Dissipation 186

- **4.4.1** Static Dissipation 188
- **4.4.2** Dynamic Dissipation 190
- **4.4.3** Low-power Design 191

4.5 Interconnect 196

- **4.5.1** Resistance 198
- **4.5.2** Capacitance 200
- **4.5.3** Delay 205
- **4.5.4** Crosstalk 207
- **4.5.5** Inductance 210
- **4.5.6** Temperature Dependence 216
- **4.5.7** An Aside on Effective Resistance and Elmore Delay 216

4.6 Wire Engineering 219

- **4.6.1** Width and Spacing 219
- **4.6.2** Layer Selection 219
- **4.6.3** Shielding 221
- **4.6.4** Repeaters 221
- **4.6.5** Implications for Logical Effort 227
- **4.6.6** Crosstalk Control 227
- **4.6.7** Low-swing Signaling 229

4.7 Design Margin 231

- **4.7.1** Supply Voltage 232
- **4.7.2** Temperature 232
- **4.7.3** Process Variation 233

4.7.4 Design Corners 233
4.7.5 Matching 235
4.7.6 Delay Tracking 237
4.8 Reliability 239
4.8.1 Reliability Terminology 239
4.8.2 Electromigration 240
4.8.3 Self-heating 241
4.8.4 Hot Carriers 241
4.8.5 Latchup 242
4.8.6 Overvoltage Failure 244
4.8.7 Soft Errors 245
4.9 Scaling 245
4.9.1 Transistor Scaling 246
4.9.2 Interconnect Scaling 249
4.9.3 International Technology Roadmap for Semiconductors 251
4.9.4 Impacts on Design 252
4.10 Pitfalls and Fallacies 258
4.11 Historical Perspective 259
Summary 264
Exercises 266

CHAPTER 5 Circuit Simulation

5.1 Introduction 273
5.2 A SPICE Tutorial 274
5.2.1 Sources and Passive Components 274
5.2.2 Transistor DC Analysis 279
5.2.3 Inverter Transient Analysis 280
5.2.4 Subcircuits and Measurement 281
5.2.5 Optimization 284
5.2.6 Other HSPICE Commands 286
5.3 Device Models 287
5.3.1 Level 1 Models 287
5.3.2 Level 2 and 3 Models 288
5.3.3 BSIM Models 288
5.3.4 Diffusion Capacitance Models 289
5.3.5 Design Corners 290
5.4 Device Characterization 292
5.4.1 I-V Characteristics 293
5.4.2 Threshold Voltage 293
5.4.3 Gate Capacitance 296

5.4.4 Parasitic Capacitance 299
5.4.5 Effective Resistance 299
5.4.6 Comparison of Processes 301
5.4.7 Process and Environmental Sensitivity 303
5.5 Circuit Characterization 303
5.5.1 Path Simulations 305
5.5.2 DC Transfer Characteristics 305
5.5.3 Logical Effort 306
5.5.4 Power and Energy 309
5.5.5 Simulating Mismatches 310
5.5.6 Monte Carlo Simulation 310
5.6 Interconnect Simulation 311
5.7 Pitfalls and Fallacies 315
Summary 316
Exercises 317

CHAPTER 6 Combinational Circuit Design

6.1 Introduction 319
6.2 Circuit Families 320
6.2.1 Static CMOS 321
6.2.2 Ratioed Circuits 327
6.2.3 Cascode Voltage Switch Logic 331
6.2.4 Dynamic Circuits 332
6.2.5 Pass-transistor Circuits 345
6.3 Circuit Pitfalls 350
6.3.1 Threshold Drops 351
6.3.2 Ratio Failures 352
6.3.3 Leakage 352
6.3.4 Charge Sharing 353
6.3.5 Power Supply Noise 353
6.3.6 Hot Spots 354
6.3.7 Minority Carrier Injection 355
6.3.8 Back-gate Coupling 356
6.3.9 Diffusion Input Noise Sensitivity 357
6.3.10 Process Sensitivity 357
6.3.11 Example: Domino Noise Budgets 357
6.4 More Circuit Families 359
6.4.1 Differential Circuits 359
6.4.2 Sense-amplifier Circuits 360

6.4.3 BiCMOS Circuits 365
6.4.4 Other Circuit Families 365
6.5 Low-power Logic Design 366
6.6 Comparison of Circuit Families 367
6.7 Silicon-on-Insulator Circuit Design 369
6.7.1 Floating Body Voltage 370
6.7.2 SOI Advantages 371
6.7.3 SOI Disadvantages 372
6.7.4 Implications for Circuit Styles 373
6.7.5 Summary 373
6.8 Pitfalls and Fallacies 374
6.9 Historical Perspective 375
Summary 377
Exercises 378

CHAPTER 7 Sequential Circuit Design

7.1 Introduction 383
7.2 Sequencing Static Circuits 384
7.2.1 Sequencing Methods 385
7.2.2 Max-Delay Constraints 388
7.2.3 Min-delay Constraints 392
7.2.4 Time Borrowing 396
7.2.5 Clock Skew 399
7.3 Circuit Design of Latches and Flip-flops 402
7.3.1 Conventional CMOS Latches 402
7.3.2 Conventional CMOS Flip-flops 405
7.3.3 Pulsed Latches 407
7.3.4 Resettable Latches and Flip-flops 408
7.3.5 Enabled Latches and Flip-flops 410
7.3.6 Incorporating Logic into Latches 410
7.3.7 Klass Semidynamic Flip-flop (SDFF) 411
7.3.8 Differential Flip-flops 412
7.3.9 True Single-phase-clock (TSPC) Latches and Flip-flops 414
7.4 Static Sequencing Element Methodology 414
7.4.1 Choice of Elements 415
7.4.2 Low-power Sequential Design 417
7.4.3 Two-phase Timing Types 418
7.4.4 Characterizing Sequencing Element Delays 422

7.5 Sequencing Dynamic Circuits 426
- **7.5.1** Traditional Domino Circuits 427
- **7.5.2** Skew-tolerant Domino Circuits 428
- **7.5.3** Unfooted Domino Gate Timing 438
- **7.5.4** Nonmonotonic Techniques 441
- **7.5.5** Static-to-domino Interface 449
- **7.5.6** Delayed Keepers 453

7.6 Synchronizers 453
- **7.6.1** Metastability 454
- **7.6.2** A Simple Synchronizer 458
- **7.6.3** Communicating Between Asynchronous Clock Domains 460
- **7.6.4** Common Synchronizer Mistakes 461
- **7.6.5** Arbiters 463
- **7.6.6** Degrees of Synchrony 464

7.7 Wave Pipelining 464

7.8 Pitfalls and Fallacies 467

7.9 Case Study: Pentium 4 and Itanium 2 Sequencing Methodologies 468
- **7.9.1** Pentium 4 Sequencing 470
- **7.9.2** Itanium 2 Sequencing 470

Summary 473

Exercises 475

CHAPTER 8 Design Methodology and Tools

8.1 Introduction 479

8.2 Structured Design Strategies 481
- **8.2.1** A Software Radio—A System Example 482
- **8.2.2** Hierarchy 485
- **8.2.3** Regularity 488
- **8.2.4** Modularity 492
- **8.2.5** Locality 495
- **8.2.6** Summary 498

8.3 Design Methods 498
- **8.3.1** Microprocessor/DSP 498
- **8.3.2** Programmable Logic 499
- **8.3.3** Gate Array and Sea of Gates Design 507
- **8.3.4** Cell-based Design 509
- **8.3.5** Full Custom Design 511
- **8.3.6** Platform-based Design—System on a Chip 518
- **8.3.7** Summary 519

8.4 Design Flows 520
8.4.1 Behavioral Synthesis Design Flow (ASIC Design Flow) 522
8.4.2 Automated Layout Generation 528
8.4.3 Mixed-signal or Custom-design Flow 532
8.4.4 Programmed Behavioral Synthesis 535
8.5 Design Economics 535
8.5.1 Non-recurring Engineering Costs (NREs) 537
8.5.2 Recurring Costs 539
8.5.3 Fixed Costs 541
8.5.4 Schedule 541
8.5.5 Personpower 542
8.5.6 Project Management 544
8.5.7 Design Reuse 544
8.6 Data Sheets and Documentation 545
8.6.1 The Summary 545
8.6.2 Pinout 546
8.6.3 Description of Operation 546
8.6.4 DC Specifications 546
8.6.5 AC Specifications 546
8.6.6 Package Diagram 547
8.6.7 Principles of Operation Manual 547
8.6.8 User Manual 547
8.7 Closing the Gap between ASIC and Custom 547
8.7.1 Microarchitecture 548
8.7.2 Sequencing Overhead 548
8.7.3 Circuit Families 549
8.7.4 Logic Design 549
8.7.5 Cell and Wire Design 550
8.7.6 Layout 550
8.7.7 Process Variation 550
8.7.8 Summary 551
8.8 CMOS Physical Design Styles 551
8.8.1 Static CMOS Gate Layout 551
8.8.2 General CMOS Layout Guidelines 553
8.8.3 Layout Optimization for Performance 556
8.9 Interchange Formats 558
8.9.1 GDS2 Stream 558
8.9.2 Caltech Intermediate Format (CIF) 558
8.9.3 Library Exchange Format (LEF) 558
8.9.4 Design Exchange Format (DEF) 559
8.9.5 Standard Delay Format (SDF) 560
8.9.6 DSPF and SPEF 561

8.9.7 Advanced Library Format (ALF) 562
8.9.8 WAVES Waveform and Vector Exchange Specification 562
8.9.9 Physical Design Exchange Format (PDEF) 563
8.9.10 OpenAccess 563
8.10 Historical Perspective 564
8.11 Pitfalls and Fallacies 564
Exercises 565

CHAPTER 9 **Testing and Verification**

9.1 Introduction 567
9.1.1 Logic Verification 568
9.1.2 Basic Digital Debugging Hints 570
9.1.3 Manufacturing Tests 573
9.2 Testers, Test Fixtures, and Test Programs 575
9.2.1 Testers and Test Fixtures 575
9.2.2 Test Programs 577
9.2.3 Handlers 579
9.3 Logic Verification Principles 579
9.3.1 Test Benches and Harnesses 579
9.3.2 Regression Testing 582
9.3.3 Version Control 584
9.3.4 Bug Tracking 584
9.4 Silicon Debug Principles 584
9.5 Manufacturing Test Principles 588
9.5.1 Fault Models 589
9.5.2 Observability 592
9.5.3 Controllability 592
9.5.4 Fault Coverage 593
9.5.5 Automatic Test Pattern Generation (ATPG) 593
9.5.6 Delay Fault Testing 594
9.6 Design for Testability 594
9.6.1 *Ad hoc* testing 595
9.6.2 Scan Design 596
9.6.3 Built-in Self-Test (BIST) 602
9.6.4 IDDQ Testing 608
9.6.5 Design for Manufacturability 608
9.7 Boundary Scan 609
9.7.1 The Test Access Port (TAP) 611
9.7.2 The Test Logic Architecture and Test Access Port 611

9.7.3 The TAP Controller 612
9.7.4 The Instruction Register 614
9.7.5 Test Data Registers 616
9.7.6 Summary 620
9.8 System-on-chip (SOC) Testing 622
9.9 Mixed-signal Testing 625
9.10 Reliability Testing 626
9.11 Testing in a University Environment 627
9.12 Pitfalls and Fallacies 629
Summary 635
Exercises 636

CHAPTER 10 Datapath Subsystems

10.1 Introduction 637
10.2 Addition/Subtraction 638
10.2.1 Single-bit Addition 638
10.2.2 Carry-propagate Addition 645
10.2.3 Adder Variants 677
10.3 One/Zero Detectors 679
10.4 Comparators 681
10.4.1 Magnitude Comparator 681
10.4.2 Equality Comparator 681
10.4.3 $K = A + B$ Comparator 682
10.5 Counters 683
10.5.1 Binary Counters 683
10.5.2 Linear-feedback Shift Registers 684
10.6 Boolean Logical Operations 686
10.7 Coding 686
10.7.1 Parity 687
10.7.2 Error-correcting Codes 687
10.7.3 Gray Codes 688
10.7.4 XOR/XNOR Circuit Forms 689
10.8 Shifters 691
10.9 Multiplication 693
10.9.1 Unsigned Array Multiplication 694
10.9.2 2's Complement Array Multiplication 696
10.9.3 Booth Encoding 698

10.9.4 Wallace Tree Multiplication 703
10.9.5 Hybrid Multiplication 705
10.9.6 Fused Multiply-Add 705
10.9.7 Serial Multiplication 705
10.10 Parallel-prefix Computations 706
10.11 Pitfalls and Fallacies 708
10.12 Historical Perspective 709
Summary 709
Exercises 710

CHAPTER 11 Array Subsystems

11.1 Introduction 713
11.2 SRAM 715
11.2.1 Memory Cell Read/Write Operation 718
11.2.2 Decoders 719
11.2.3 Bitline Conditioning and Column Circuitry 725
11.2.4 Multi-ported SRAM and Register Files 728
11.2.5 Large SRAMs 730
11.2.6 Logical Effort of RAMs and Register Files 731
11.2.7 Case Study: Itanium 2 Cache 733
11.3 DRAM 734
11.3.1 Subarray Architectures 735
11.3.2 Column Circuitry 738
11.3.3 Applications to CMOS Systems-on-chip 738
11.4 Read-only Memory 739
11.4.1 Programmable ROMs 741
11.4.2 NAND ROMs 743
11.5 Serial Access Memories 744
11.5.1 Shift Registers 745
11.5.2 Queues (FIFO, LIFO) 746
11.6 Content-addressable Memory 747
11.7 Programmable Logic Arrays 750
11.8 Array Yield, Reliability, and Self-test 756
11.9 Historical Perspective 757
Summary 759
Exercises 760

CHAPTER 12 Special-purpose Subsystems

12.1 Introduction 761

12.2 Packaging 761

- **12.2.1** Package Options 762
- **12.2.2** Chip-to-package Connections 764
- **12.2.3** Package Parasitics 765
- **12.2.4** Heat Dissipation 765

12.3 Power Distribution 767

- **12.3.1** On-chip Power Distribution Network 768
- **12.3.2** IR Drops 771
- **12.3.3** L di/dt Noise 772
- **12.3.4** On-chip Bypass Capacitance 773
- **12.3.5** Power Network Modeling 775
- **12.3.6** Signal Return Paths 778
- **12.3.7** Power Supply Filtering 779
- **12.3.8** Substrate Noise 780

12.4 I/O 780

- **12.4.1** Basic I/O Pad Circuits 781
- **12.4.2** Example: MOSIS I/O Pads 784
- **12.4.3** Level Converters 784

12.5 Clock 786

- **12.5.1** Definitions 786
- **12.5.2** Clock System Architecture 789
- **12.5.3** Global Clock Generation 790
- **12.5.4** Global Clock Distribution 793
- **12.5.5** Local Clock Gaters 798
- **12.5.6** Clock Skew Budgets 800
- **12.5.7** Adaptive Deskewing 806
- **12.5.8** Clocking Alternatives 807

12.6 Analog Circuits 808

- **12.6.1** MOS Small-signal Model 808
- **12.6.2** Common Source Amplifier 811
- **12.6.3** The CMOS Inverter as an Amplifier 812
- **12.6.4** Current Mirrors 814
- **12.6.5** Differential Pairs 816
- **12.6.6** Simple CMOS Operational Amplifier 819
- **12.6.7** Digital-to-analog and Analog-to-digital Converter Basics 819
- **12.6.8** Digital-to-analog Converters 824
- **12.6.9** Analog-to-digital Converters 828
- **12.6.10** Radio Frequency (RF) Circuits 837
- **12.6.11** Analog Summary 841

12.7 Pitfalls and Fallacies 842
12.8 Historical Perspective 843
Summary 845
Exercises 845

APPENDIX A Verilog

A.1 Introduction 849
A.2 Behavioral Modeling with Continuous Assignments 850
A.2.1 Bitwise Operators 850
A.2.2 Comments and White Space 851
A.2.3 Reduction Operators 851
A.2.4 Other Operators 852
A.3 Basic Constructs 852
A.3.1 Internal Signals 852
A.3.2 Precedence 853
A.3.3 Constants 854
A.3.4 Hierarchy 854
A.3.5 Tristates 855
A.3.6 Bit Swizzling 855
A.3.7 Delays 857
A.4 Behavioral Modeling with Always Blocks 857
A.4.1 Registers 857
A.4.2 Latches 859
A.4.3 Counters 859
A.4.4 Combinational Logic 860
A.4.5 Memories 866
A.4.6 Blocking and Nonblocking Assignment 867
A.5 Finite State Machines 868
A.6 Parameterized Modules 874
A.7 Structural Primitives 874
A.8 Test Benches 875
A.9 Pitfalls 877
A.9.1 Verilog Style Guidelines 877
A.9.2 Incorrect Stimulus List 878
A.9.3 Missing `begin/end` Block 880
A.9.4 Undefined Outputs 880
A.9.5 Incomplete Specification of Cases 882

A.9.6 Shorted Outputs 884
A.9.7 Incorrect Use of Nonblocking Assignments 885
A.10 Example: MIPS Processor 886

APPENDIX B VHDL

B.1 Introduction 895
B.2 Behavioral Modeling with Concurrent Signal Assignments 896
B.2.1 Bitwise Operators 896
B.2.2 Comments and White Space 897
B.2.3 Other Operators 897
B.2.4 Conditional Signal Assignment Statements 898
B.2.5 Selected Signal Assignment Statements 898
B.3 Basic Constructs 899
B.3.1 Blocks, Entities, and Architectures 899
B.3.2 Internal Signals 900
B.3.3 Precedence 900
B.3.4 Hierarchy 901
B.3.5 Bit Swizzling 901
B.3.6 Types 902
B.3.7 Library and Use Clauses 904
B.3.8 Tristates 905
B.3.9 Delays 906
B.4 Behavioral Modeling with Process Statements 906
B.4.1 Flip-flops 906
B.4.2 Latches 908
B.4.3 Counters 908
B.4.4 Combinational Logic 909
B.4.5 Memories 912
B.5 Finite State Machines 913
B.6 Parameterized Blocks 915
B.7 Example: MIPS Processor 917

References 927

Index 953

Preface

In the two decades since the first edition of this book was published, CMOS technology has claimed the preeminent position in modern electrical system design and enabled the widespread use of personal computers. In the decade since the second edition was published, continued advances have led to the explosion of the Internet and wireless communications. The transistor counts and clock frequencies of state-of-the-art chips have grown by orders of magnitude.

	1st Edition	2nd Edition	3rd Edition
Year	1985	1993	2004
Transistor Counts	10^5–10^6	10^6–10^7	10^8–10^9
Clock Frequencies	10^7	10^8	10^9
Worldwide Market	$25B	$60B	$170B

This book has been extensively rewritten to reflect the enormous advances in integrated circuit design over the past decade. While the basic principles are largely the same, the practices have changed enormously because of the increases in transistor budgets and clock speeds, the growing challenges of power consumption, and the improvements in productivity and CAD tools.

How to Use This Book

This book intentionally covers more breadth and depth than any course would cover in a semester. It is accessible for a first undergraduate course in VLSI, yet detailed enough for advanced graduate courses and is useful as a reference to the practicing engineer. You are encouraged to pick and choose topics according to your interest. Chapter 1 previews the entire field, while subsequent chapters elaborate on specific topics. Sections are marked as optional (see example icon in margin) if they are not needed to understand subsequent sections. You may skip them on a first reading and return when they are relevant to you.



We have endeavored to include figures whenever possible ("a picture is worth a thousand words") to trigger your thinking. As you encounter examples throughout the text, we urge you to think about them before reading on to the solutions. We have also provided

extensive references for those who need to delve deeper into topics introduced in this text. We have emphasized the best practices that are used in industry and warned of pitfalls and fallacies. Our judgments about merits of circuits may become incorrect as technology and applications change, but we believe it is the responsibility of a writer to attempt to separate the good from the bad.

Supplements

Updating and expanding on the supplements available was a major goal of this edition. There is now a volume of support materials for students and professors. All these are available on the Companion Web site for the book, `www.aw-bc.com/weste`. Supplements to help students with the course include:

- A lab manual with laboratory exercises involving the design of an 8-bit microprocessor covered in Chapter 1.
- A collection of links to VLSI resources including open-source CAD tools and process parameters.
- A student solutions manual that includes answers to selected problems.

Supplements to help instructors with the course include:

- A sample syllabus.
- Lecture slides for an introductory VLSI course.
- An instructors' manual with solutions.

These materials have been prepared exclusively for professors using the book in a course. Please contact your local Addison-Wesley sales representative, or send email to `aw.cse@aw.com`, for information on how to access them.

Acknowledgments

I (Neil) would like to first acknowledge my coauthor David Harris, who convinced me to come out of "book retirement" to do this edition. Without his tireless efforts, this book would not exist. Next, my long-suffering spouse, Avril, who relented on "Never another book!"—one more time.

Over the past ten or so years, along with my colleagues at Radiata Communications / Cisco Systems, I have learned a lot about getting mixed signal and RF CMOS circuits from conception into production. Much of that experience is embedded in this new edition. Gordon Foyster wrote the software that allowed us to convert mask layouts to postscript, a task that hasn't eased in arcaneness in ten years. Steve Avery helped out with

suggestions on how to deal with the mask artwork and various other "book savvy" comments. Geoff Smith did the synthesis of the place and route examples that appear in Chapter 1, and along with Gordon often provided design methodology suggestions. Jared Anderson did the MATLAB and synthesis of the NCO in Chapter 8. Phil Ryan, Greg Zyner, and Mike Webb provided background on digital methodologies and design management. Andrew Adams, Jeffrey Harrison and John Olip guided me in RF issues. Rodney Chandler provided ADC theory and practice. Brian Hart contributed the INL/DNL plots in Chapter 12. Tom McDermott often provided software guidance and other eclectic information. Chris Corcoran often got me out of a muddle with dead laptops or other seemingly insurmountable software or network problems. John O'Sullivan provided useful feedback and was a good person to bounce ideas off. George Bouchaya brought an extensive background in manufacturing. John Haddy provided some of the photographs in this edition.

Bronwyn Forde provided administrative assistance. Dave Leonard and Bill Rossi provided support from afar. Finally, I would like to thank my long-time business partners and friends, Dave Skellern, Chris Beare, and Don MacLennan for a fruitful and productive association. I would like to acknowledge Cisco Systems in general for its support of this revision.

In contrast to the second edition, which was completed on an Apple Macintosh and a Symbolics Ivory processor in a basement in Massachusetts, many of my contributions to this edition were completed using the very technology that has become pervasive in the last ten or so years. Wireless hotspots and ADSL at home and in hotels and airports all over Australia and the world played their part in this edition. And both PC and Mac laptop technology played a huge part in the revision—providing the ability to work anywhere, anytime.

I (David) owe a great deal to the outstanding circuit designers with whom I have worked over the years. Mark Horowitz, Jonathan Allen, Bill Dally, Ivan Sutherland, Jason Stinson, Sam Naffziger, Tom Fletcher, and the Horowitz group at Stanford have shaped the way I think about circuits. I hope this book passes on some of the insights they have taught me. I would also like to thank Peter Cheung of Imperial College for hosting me during a very productive summer of writing in London.

We are indebted to many people for their reviews and suggestions. These people include: Bharadwaj "Birdy" Amrutur, Jacob Baker, Kerry Bernstein, Neil Burgess, Krishnendu Chakrabarty, C. K. Chen, Bill Dally, Nana Dankwa, Azita Emami-Neyestanak, Scott Fairbanks, Tom Fletcher, Jim Frenzel, Claude Gauthier, Ron Ho, David Hopkins, Nan "Ted" Jiang, Marcie Karty, Stephen Keckler, Fabian Klass, Torsten Lehmann, Rich Lethin, Michael Linderman, Dean Liu, Wagdy Mahmoud, Ziyad Mansour, Simon Moore, Alice Parker, Braden Phillips, Parameswaran Ramanathan, Justin Schauer, Ashok Srivastava, James Stine, Gu Wei, Ken Yang, and Evelina Yeung. Jaeha Kim, Tom Grutkowski, and Cecilia Krasuk provided particularly thorough technical reviews of large portions of the manuscript. We apologize in advance to anyone we overlooked.

TSMC kindly provided permission to use the 180 nm SPICE models for many examples. The MOSIS Service has provided measured SPICE parameters for numerous other processes. Artisan contributed data sheet pages from a TSMC 180 nm cell library.

Harvey Mudd College provided chip photographs taken by Kevin Mapp. Steve Rubin developed the open-source Electric Editor used to produce many layouts.

Designers from several companies contributed "war stories" of chip design errors for Section 9.12 under the condition of anonymity. We welcome your war stories as candidates for the next edition. An ideal story would be presented like a detective mystery starting with the symptoms, then the process by which the bug was traced, and concluding with a schematic of the incorrect circuit and the method by which it was corrected.

The E158 CMOS VLSI Design classes of Spring 2002, 2003, and 2004 at Harvey Mudd College and classes of engineers at Qualcomm and Sun Microsystems tested drafts of the manuscript. Some of the engineers who helped improve the manuscript include Matt Aldrich, Kevin Alley, Chi Bui, Ayoob Dooply, Trevor Gile, Brad Greer, Shamit Grover, Eric Henderson, Nick Hertl, Nicole Kang, Clark Korb, Karen Lee, Li-Jen Lin, Michael Linderman, Mark Locascio, Renee Logan, Dimitrios Lymberopoulos, Khurram Malik, Charles Matlack, Joe Petolino, Geoff Shippee, Joshua Smallman, Keith Stevens, Aaron Stratton, Yushi Tian, Daniel Woo, and Amy Yang.

Genevieve Breed, Matthew Erler, Tommy Leung, and David Diaz from Harvey Mudd College developed many of the simulations and figures throughout the text. David Diaz, Sean Kao, and Daniel Lee helped develop the MIPS processor example at Harvey Mudd College. Max Yi ported the MIPS example from Appendix B to Appendix A.

Addison-Wesley has done an admirable job with the grueling editorial and production process. We would particularly like to thank our editors Maite Suarez-Rivas, Matt Goldstein, and Juliet Silveri, our copyeditor Kathy Smith, and our compositor and artist Gillian Hall.

Sally Harris has been editing family books since David was an infant on her lap. She read the page proofs with amazing attention to detail and unearthed hundreds of errors. She (with help from Daniel Harris) also volunteered for the thankless task of checking the bibliography.

We have become painfully aware of the ease with which mistakes creep into a book of this size. The remaining errors are our own fault. We will pay a bounty of $1 to the first person to report each error, paid at the next printing. Please check the errata sheet at `www.aw-bc.com/weste` to see if the bug has already been reported. Send your reports with your name and address to `bugs@cmosvlsi.com`.

N.W.
D.H.
April 2004

Introduction 1

1.1 A Brief History

In 1958, Jack Kilby built the first integrated circuit flip-flop with two transistors at Texas Instruments. In 2003, the Intel Pentium 4 microprocessor contained 55 million transistors and a 512-Mbit dynamic random access memory (DRAM) contained more than half a billion transistors. This corresponds to a compound annual growth rate of 53% over 45 years. No other technology in history has sustained such a high growth rate for so long.

This incredible growth has come from steady miniaturization of transistors and improvements in manufacturing processes. Most other fields of engineering involve tradeoffs between performance, power, and price. However, as transistors become smaller, they also become faster, dissipate less power, and are cheaper to manufacture. This synergy has revolutionized not only electronics, but also society at large.

The processing performance once exclusive to Cray supercomputers is now available in hand-held personal digital assistants. Processing capability that was once necessary for secret military spread-spectrum communications is now available in disposable cellular telephones. Improvements in integrated circuits have enabled space exploration, made automobiles more efficient, revolutionized the nature of warfare, brought vast libraries of information to our Web browsers, and made the world a more interdependent place.

Figure 1.1 shows annual sales in the worldwide semiconductor market. Integrated circuits became a $100B/year business in 1994. The blip in 2000 is associated with the surge in sales for Y2K upgrades followed by the worldwide recession. In 2003, the industry manufactured more than one quintillion (10^{18}) transistors, or more than 100 million for every human being on the planet. Thousands of engineers have made their fortunes in the field. New fortunes lie ahead for those with innovative ideas and the talent to bring their ideas to reality.

During the first half of the 20th century, electronic circuits used large, expensive, power-hungry, and unreliable vacuum tubes. In 1947, John Bardeen and Walter Brattain built the first functioning point contact transistor at Bell Laboratories, shown in Figure 1.2(a) [Riordan97]. It was nearly classified as a military secret, but Bell Labs publicly announced the device in the following year.

> *We have called it the Transistor, T-R-A-N-S-I-S-T-O-R, because it is a resistor or semiconductor device which can amplify electrical signals as they are transferred*

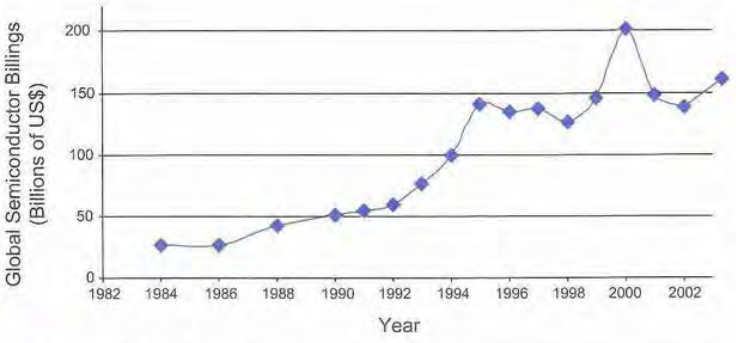


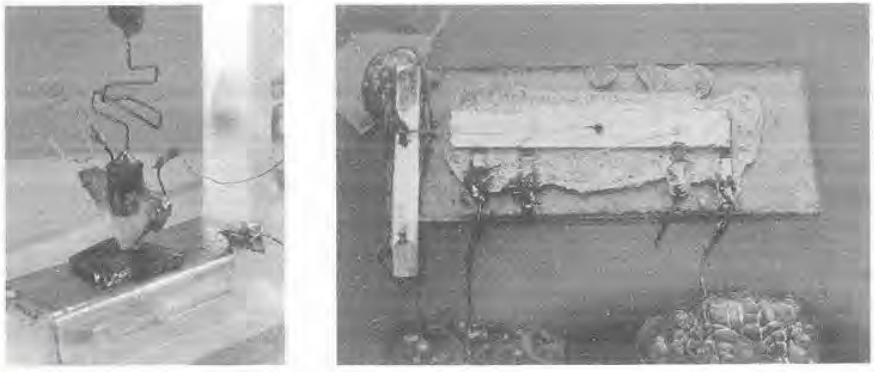
FIG 1.1 Size of worldwide semiconductor market

Source: Semiconductor Industry Association.

through it from input to output terminals. It is, if you will, the electrical equivalent of a vacuum tube amplifier. But there the similarity ceases. It has no vacuum, no filament, no glass tube. It is composed entirely of cold, solid substances.

Ten years later, Jack Kilby at Texas Instruments realized the potential for miniaturization if multiple transistors could be built on a single piece of silicon. Figure 1.2(b) shows his first prototype of an integrated circuit, constructed from a germanium slice and gold wires.

The invention of the transistor earned the Nobel Prize in Physics in 1956 for Bardeen, Brattain, and their co-worker William Shockley. Kilby received the Nobel Prize in Physics in 2000 for the invention of the integrated circuit.

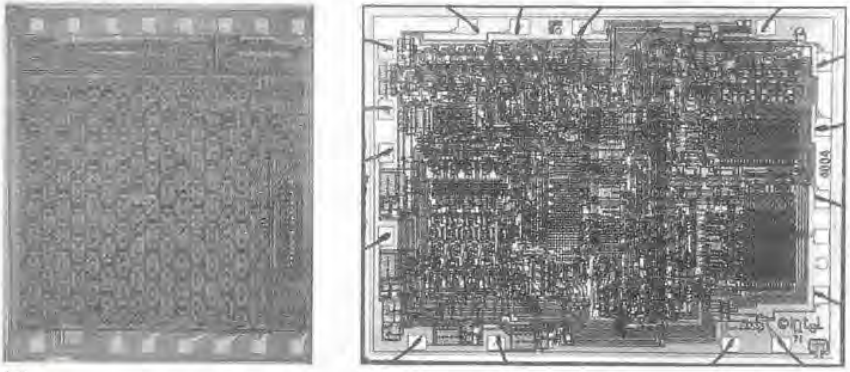


(a) (b)

FIG 1.2 (a) First transistor (Courtesy of Texas Instruments.) and (b) first integrated circuit. (Property of AT&T Archives. Reprinted with permission of AT&T.)

Soon after inventing the point contact transistor, Bell Labs developed the bipolar junction transistor. Bipolar transistors were more reliable, less noisy, and more power-efficient. Early integrated circuits primarily used bipolar transistors. Transistors can be viewed as electrically controlled switches with a control terminal and two other terminals that are connected or disconnected depending on the voltage applied to the control. Bipolar transistors require a small current into the control (base) terminal to switch much larger currents between the other two (emitter and collector) terminals. The quiescent power dissipated by these base currents limits the maximum number of transistors that can be integrated onto a single die. Metal Oxide Semiconductor Field Effect Transistors (MOSFETs) offer the compelling advantage that they draw almost zero control current while idle. They come in two flavors: nMOS and pMOS, using n-type and p-type dopants, respectively. The original idea of field effect transistors dated back to the German scientist Julius Lilienfield in 1925 [US patent 1,745,175] and a structure closely resembling the MOSFET was proposed in 1935 by Oskar Heil [British patent 439,457], but materials problems foiled early attempts to make functioning devices.

Frank Wanlass at Fairchild described the first logic gates using MOSFETs in 1963 [Wanlass63]. His gates used both nMOS and pMOS transistors, earning the name Complementary Metal Oxide Semiconductor, or CMOS. The circuits used discrete transistors but consumed only nanowatts of power, six orders of magnitude less than their bipolar counterparts. With the development of the silicon planar process, MOS integrated circuits became attractive for their low cost because each transistor occupied less area and the fabrication process was simpler [Vadasz69]. Early processes used only pMOS transistors and suffered from poor performance, yield, and reliability. Processes using nMOS transistors became dominant in the 1970s [Mead80]. Intel pioneered nMOS technology with its 1101 256-bit static random access memory and 4004 4-bit microprocessor shown in Figure 1.3. While the nMOS process was less expensive than CMOS, nMOS logic gates still consumed power while idle. Power consumption became a major issue in the 1980s as



(a) (b)

FIG 1.3 (a) Intel 1101 SRAM (© IEEE 1967 [Vadasz69]) and (b) 4004 microprocessor (Reprinted with permission of Intel Corporation.)

hundreds of thousands of transistors were integrated onto a single die. CMOS processes were widely adopted and have essentially replaced nMOS and bipolar processes for nearly all digital logic applications.

Gordon Moore observed in 1965 that plotting the number of transistors that can be most economically manufactured on a chip gives a straight line on a semilogarithmic scale [Moore65]. At the time he found transistor count doubling every 18 months. This observation has been called *Moore's Law* and has become a self-fulfilling prophecy. Figure 1.4 shows that the number of transistors in Intel microprocessors has doubled every 26 months since the invention of the 4004.

The level of integration of chips has been classified as small-scale, medium-scale, large-scale, and very large-scale. *Small-scale integration* (SSI) circuits such as the 7404 inverter have fewer than 10 gates, with a conversion of roughly half a dozen transistors per gate. *Medium-scale integration* (MSI) circuits such as the 74161 counter have up to 1000 gates. *Large-scale integration* (LSI) circuits such as simple 8-bit microprocessors have up to 10,000 gates. It soon became apparent that new names would have to be created every five years if this naming trend continued and thus the term *very large-scale integration* (VLSI) is used to describe most integrated circuits from the 1980s onward.

A corollary of Moore's law is that transistors become faster, consume less power, and are cheaper to manufacture each year. Figure 1.5 shows that Intel microprocessor clock frequencies have doubled roughly every 34 months. Remarkably, the improvements have accelerated in recent years. Computer performance has grown even more than raw clock speed. Even though an individual CMOS transistor uses very little energy each time it switches, the enormous numbers of transistors switching at very high rates of speed have made power consumption a major design consideration again.

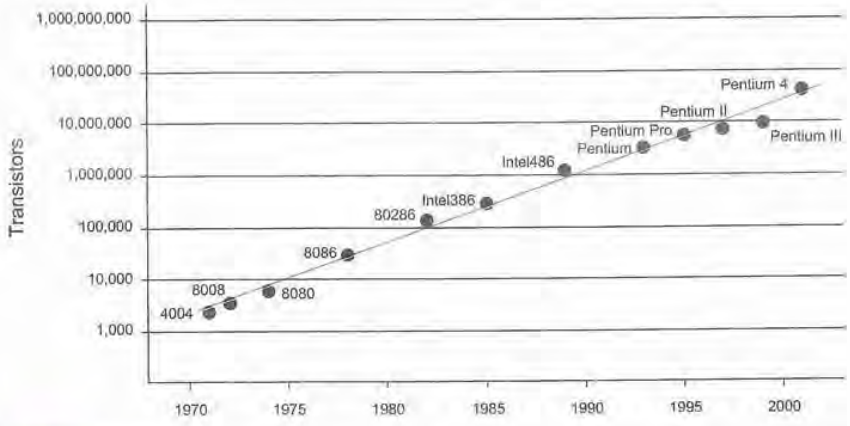


FIG 1.4 Transistors in Intel microprocessors [Intel03]

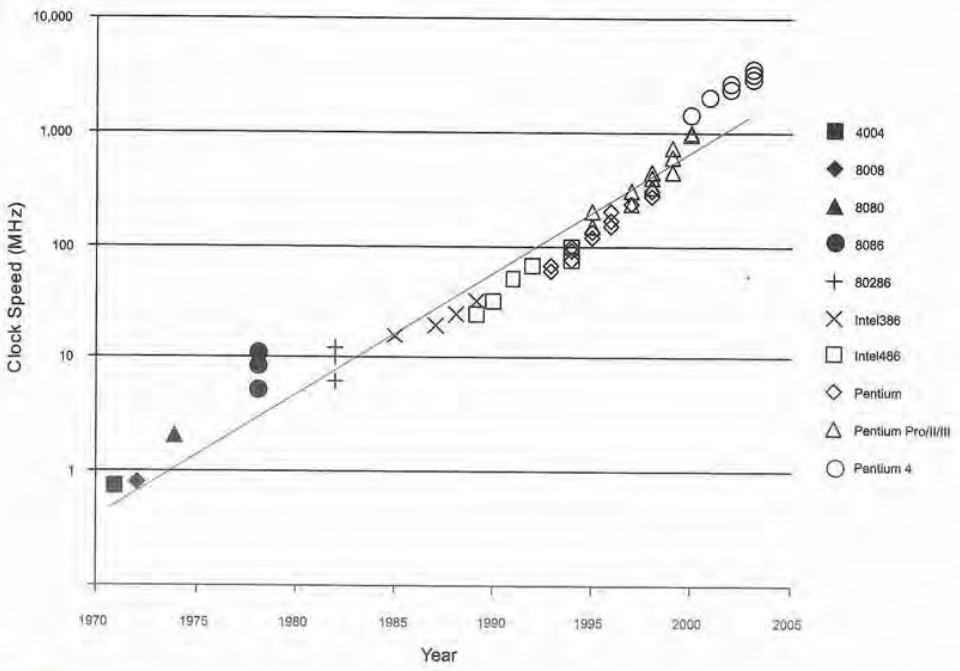


FIG 1.5 Clock frequencies of Intel microprocessors

The 4004 used transistors with minimum dimensions of 10 μm in 1971. The Pentium 4 uses transistors with minimum dimensions of 130 nm in 2003, corresponding to two orders of magnitude in improvement over three decades. Obviously this *scaling* cannot go on forever because transistors cannot be smaller than atoms. However, many predictions of fundamental limits to scaling have already proven wrong. Creative engineers and material scientists have billions of dollars to gain by getting ahead of their competitors. In the early 1990s, experts agreed that scaling would continue for at least a decade but that beyond that point the future was murky. In 2003, we still believe that scaling will continue for at least another decade. The future is yours to invent.

1.2 Book Summary

As VLSI transistor budgets have grown exponentially, designers have come to rely on increasing levels of automation to seek corresponding productivity gains. Many designers spend much of their effort specifying circuits with hardware description languages and sel-

dom look at actual transistors. Nevertheless, chip design is not software engineering. Addressing the harder problems requires a fundamental understanding of circuit and physical design. Therefore, this book focuses on building an understanding of integrated circuits from the bottom up.

In this chapter we will take a simplified view of CMOS transistors as switches. With this model we will develop CMOS logic gates and latches. CMOS transistors are mass-produced on silicon wafers using lithographic steps much like a printing press process. We will explore how to lay out transistors by specifying rectangles indicating where dopants should be diffused, polysilicon should be grown, metal wires should be deposited, and contacts should be etched to connect all the layers together. By the middle of this chapter you will understand all the principles required to design and lay out your own simple CMOS chip. The chapter concludes with an extended example demonstrating the design of a simple 8-bit MIPS microprocessor chip. The processor raises many of the design issues that will be developed in more depth throughout the book. The best way to learn VLSI design is by doing it. A set of laboratory exercises are available on the Web (see the Web Supplement portion of the preface) to guide you through the design of your own microprocessor chip.

Of course, transistors are not simply switches. Chapter 2 develops first-order current-voltage (I-V) and capacitance-voltage (C-V) models for transistors and describes the important second-order effects. These models are used to predict the transfer characteristics of CMOS inverters. A summary of CMOS processing technology is presented in Chapter 3. The basic processes in current use are described along with some interesting process enhancements. Representative layout design rules are also presented. Chapter 4 addresses performance estimation for circuits. The first-order I-V characteristics are both too complicated and too inaccurate to apply by hand to interesting circuits. Fortunately, transistors can be modeled as having an effective resistance and capacitance for the purpose of estimating delay. The relative performance of different gates can be quantified through their logical effort, a technique that we will revisit throughout the book. Wires are also addressed because they are as important as the transistors to overall performance. Simulation is discussed in Chapter 5 and is used to obtain more accurate performance predictions as well as to verify the correctness of circuits and logic. Chapter 6 addresses combinational circuit design. A whole kit of circuit families are available with different tradeoffs in speed, power, complexity, and robustness. Chapter 7 continues with sequential circuit design, including clocking and latching techniques.

The remainder of this book presents a subsystem view of CMOS design. Chapter 8 focuses on a range of current design methods, identifying the issues peculiar to CMOS. Testing and design-for-test techniques are discussed in Chapter 9. Chapter 10 catalogs designs for a host of datapath subsystems including adders, shifters, multipliers, counters, and others. Chapter 11 similarly describes memory subsystems including SRAMs, DRAMs, CAMs, ROMs, and PLAs. Finally, Chapter 12 addresses special-purpose subsystems including clocking, I/O, and mixed-signal blocks. Hardware description languages (HDLs) are used in the design of nearly all digital integrated circuits today. Appendices A and B provide tutorials in Verilog and VHDL, the two dominant HDLs.

A number of sections are marked with an "optional" icon. These sections describe particular subjects in greater detail. You may skip over these sections on a first reading and return to them when they are of practical relevance.



1.3 MOS Transistors

Silicon (Si), a semiconductor, forms the basic starting material for a large class of integrated circuits [Tsividis99]. Pure silicon consists of a three-dimensional *lattice* of atoms. Silicon is a Group IV element, so it forms covalent bonds with four adjacent atoms, as shown in Figure 1.6(a). The lattice is shown in the plane for ease of drawing, but it actually forms a cubic crystal. As all of its valence electrons are involved in chemical bonds, it is a poor conductor. The conductivity can be raised by introducing small amounts of impurities into the silicon lattice. These impurities are called *dopants*. A Group V dopant such as arsenic has five valence electrons. It replaces a silicon atom in the lattice and still bonds to four neighbors, so the fifth valence electron is loosely bound to the arsenic atom, as shown in Figure 1.6(b). Thermal vibration of the lattice at room temperature is enough to set the electron free to move, leaving a positively charged As^+ ion and a free electron. The free electron can carry current so the conductivity is higher. We call this an *n*-type semiconductor because the free carriers are negatively charged electrons. Similarly, a Group III dopant such as boron has three valence electrons, as shown in Figure 1.6(c). The dopant atom can borrow an electron from a neighboring silicon atom, which in turn becomes short by one electron. That atom in turn can borrow an electron, and so forth, so the missing electron, or *hole*, can propagate about the lattice. The hole acts as a positive carrier so we call this a *p*-type semiconductor.

A junction between p-type and n-type silicon is called a *diode*, shown in Figure 1.7. When the voltage on the p-type semiconductor, called the *anode*, is raised above the n-type *cathode*, we say the diode is *forward biased* and current will flow. When the anode voltage is less than or equal to the cathode voltage, the diode is *reverse biased* and almost zero current flows.

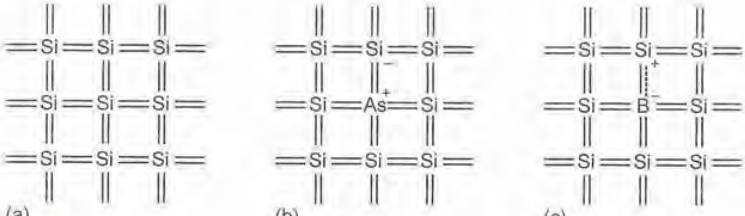


FIG 1.6 Silicon lattice and dopant atoms

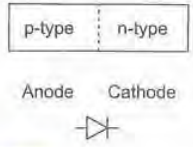


FIG 1.7 p-n junction diode structure and symbol

An *MOS* (Metal-Oxide-Semiconductor) structure is created by superimposing several layers of conducting and insulating materials to form a sandwich-like structure. These structures are manufactured using a series of chemical processing steps involving oxidation of the silicon, the diffusion of impurities into the silicon to give it certain conduction characteristics, and the deposition and etching of aluminum or other metals to provide interconnection in the same way that a printed wiring board is constructed. This is carried out on a single crystal of silicon, which is available as thin flat circular wafers around 15–30 cm in diameter. CMOS technology provides two types of transistors (also called *devices* in this text): an n-type transistor (nMOS) and a p-type transistor (pMOS). Transistor operation is based on electric fields so the devices are also called Metal Oxide Semiconductor Field Effect Transistors (*MOSFETs*) or simply *FETs*. Cross-sections and symbols of these transistors are shown in Figure 1.8. The n+ and p+ regions indicate heavily doped n- or p-type silicon.

Each transistor consists of a stack of the conducting *gate*, an insulating layer of silicon dioxide (SiO_2, better known as glass), and the silicon wafer, also called the *substrate*, *body*, or *bulk*. Gates of early transistors were built from metal, so the stack was called metal-oxide-semiconductor, or MOS. Now the gate is typically formed from polycrystalline silicon (*polysilicon*), but the name stuck. An nMOS transistor is built with a p-type body and has regions of n-type semiconductor adjacent to the gate called the *source* and *drain*. They are physically equivalent and for now we will regard them as interchangeable. The body is typically grounded. A pMOS transistor is just the opposite, consisting of p-type source and drain regions with an n-type body. In a CMOS technology with both flavors of transistors, the substrate is either n-type or p-type. The other flavor of transistor must be built in a special well in which dopant atoms have been locally added to form the body of the opposite type.

The gate is a control input: It affects the flow of electrical current between the source and drain. Consider an nMOS transistor. The body is generally grounded so the p–n junc-

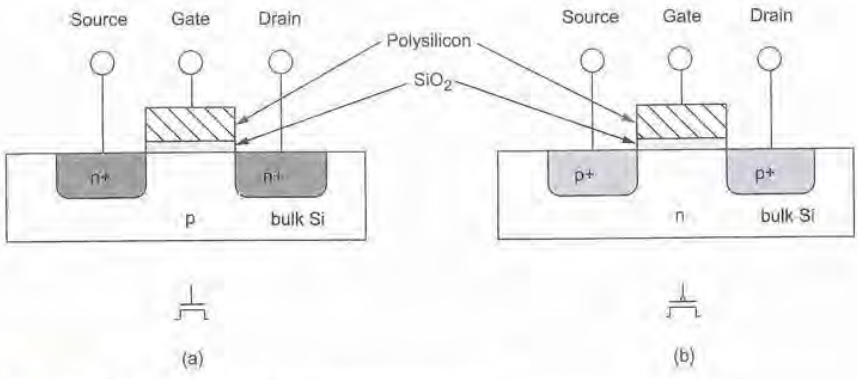


FIG 1.8 nMOS transistor (a) and pMOS transistor (b)

tions of the source and drain to body are reverse-biased. If the gate is also grounded, no current flows through the reverse-biased junctions. Hence, we say the transistor is OFF. If the gate voltage is raised, it creates an electric field that starts to attract free electrons to the underside of the Si–SiO_2 interface. If the voltage is raised enough, the electrons outnumber the holes and a thin region under the gate called the *channel* is inverted to act as an n-type semiconductor. Hence, a conducting path of electron carriers is formed from source to drain and current can flow. We say the transistor is ON.

For a pMOS transistor, the situation is again reversed. The body is held at a high potential. When the gate is also at a high potential, the source and drain junctions are reverse-biased and no current flows so the transistor is OFF. When the gate voltage is lowered, positive charges are attracted to the underside of the Si–SiO_2 interface. A sufficiently low gate voltage inverts the channel and a conducting path of positive carriers is formed from source to drain, so the transistor is ON. Notice that the symbol for the pMOS transistor has a bubble on the gate, indicating that the transistor behavior is the opposite of the nMOS.

Throughout this book, we will call the high potential V_{DD} or POWER and assume it represents a logic '1' value in digital circuits. In many logic families of the 1970s and 1980s, V_{DD} was set to 5 volts or occasionally higher. Smaller, more recent transistors are unable to withstand such high voltages and have used supplies of 3.3 V, 2.5 V, 1.8 V, 1.5 V, and so forth. The low potential is called GROUND (GND) or V_{SS} and represents a logic '0.' It is normally 0 volts.

In summary, the gate of an MOS transistor controls the flow of current between the source and drain. Simplifying this to the extreme allows the MOS transistors to be viewed as simple on/off switches. When the gate of an nMOS transistor is '1,' the transistor is ON and there is a conducting path from source to drain. When the gate is low, the nMOS transistor is OFF and almost zero current flows from source to drain. A pMOS transistor is just the opposite, being ON when the gate is low and OFF when the gate is high. This switch model is illustrated in Figure 1.9, where *g*, *s*, and *d* indicate gate, source, and drain. It is so useful that it will be our most common model for thinking about circuit behavior.

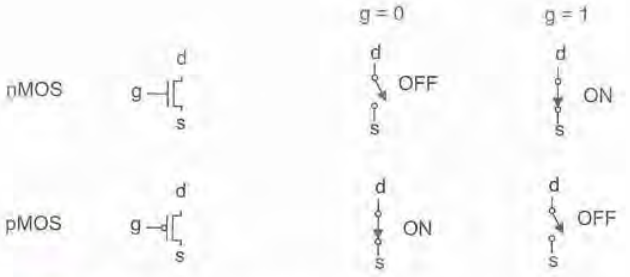


FIG 1.9 Transistor symbols and switch-level models

1.4 CMOS Logic

1.4.1 The Inverter

Figure 1.10(a) shows a CMOS inverter or NOT gate using one nMOS transistor and one pMOS transistor. The horizontal bar at the top indicates V_{DD} and the triangle at the bottom indicates GND. When the input A is '0,' the nMOS transistor is OFF and the pMOS transistor is ON. Thus the output Y is pulled up to '1' because it is connected to V_{DD} but not to GND. Conversely, when A is '1,' the nMOS is ON, the pMOS is OFF, and the Y is pulled down to '0.' This is summarized in the truth table of Table 1.1. The symbol is given in Figure 1.10(b).

Table 1.1 Inverter truth table

A	Y
0	1
1	0

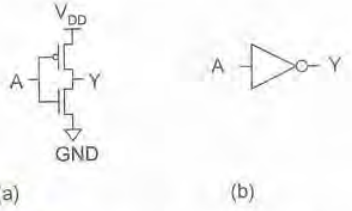


FIG 1.10 Inverter schematic (a) and symbol (b) $Y = \overline{A}$

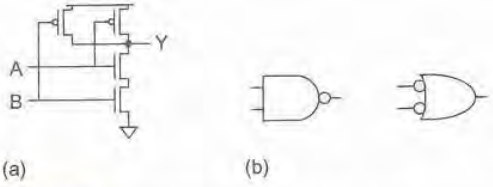


FIG 1.11 2-input NAND gate schematic (a) and symbol (b) $Y = \overline{A \bullet B}$

1.4.2 The NAND Gate

Figure 1.11(a) shows a 2-input CMOS NAND gate. It consists of two series nMOS transistors between Y and GND and two parallel pMOS transistors between Y and V_{DD}. If either input A or B is '0,' at least one of the nMOS transistors will be OFF, breaking the path from Y to GND. But at least one of the pMOS transistors will be ON, creating a path from Y to V_{DD}. Hence, the output Y will be '1.' If both inputs are '1,' both of the nMOS transistors will be ON and both of the pMOS transistors will be OFF. Hence, the output will be '0.' The truth table is given in Table 1.2 and the symbol is shown in Figure 1.11(b). Note that by DeMorgan's Law, the inversion bubble may be placed on either side of the gate. In the figures in this book, two lines intersecting at a T-junction are connected. Two lines crossing are connected if and only if a dot is shown.

k-input NAND gates are constructed using k series nMOS transistors and k parallel pMOS transistors. For example, a 3-input NAND gate is shown in Figure 1.12. When any of the inputs are 0, the output is pulled high through the parallel pMOS transistors. When all of the inputs are '1,' the output is pulled low through the series nMOS transistors.

Table 1.2 NAND gate truth table

A	B	pull-down network	pull-up network	Y
0	0	OFF	ON	1
0	1	OFF	ON	1
1	0	OFF	ON	1
1	1	ON	OFF	0

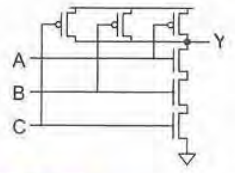


FIG 1.12 3-input NAND gate schematic
$Y = \overline{A \bullet B \bullet C}$

1.4.3 Combinational Logic

The inverter and NAND gates are examples of *complementary CMOS logic gates*, also called *static CMOS gates*. In general, a fully complementary CMOS gate has an nMOS *pull-down network* to connect the output to '0' (GND) and pMOS *pull-up network* to connect the output to '1' (V_{DD}), as shown in Figure 1.13. The networks are arranged such that one is ON and the other OFF for any input pattern.

The pull-up and pull-down networks in the inverter each consisted of a single transistor. The NAND gate used a series pull-down network and a parallel pull-up network. More elaborate networks are used for more complex gates. Two or more transistors in series are ON only if all of the series transistors are ON. Two or more transistors in parallel are ON if any of the parallel transistors are ON. This is illustrated in Figure 1.14 for nMOS and pMOS transistor pairs. By using combinations of these constructions, CMOS combinational gates can be constructed.

In general when we join a pull-up network to a pull-down network to form a logic gate as shown in Figure 1.13, they both will attempt to exert a logic level at the output. The possible levels at the output are shown in Table 1.3. From this table it can be seen that the output of a CMOS logic gate can be in four states. The '1' and '0' levels have been encountered with the inverter and NAND gates, where either the pull-up or pull-down is OFF and the other structure is ON. When both pull-up and pull-down are OFF, the *high-impedance* or *floating* Z output state results. This is of importance in multiplexers, memory elements, and bus drivers. The *crowbarred* X level exists when both pull-up and pull-down are simultaneously turned ON. This causes an indeterminate level and also static power to be dissipated. It is usually an unwanted condition in any CMOS digital circuit.

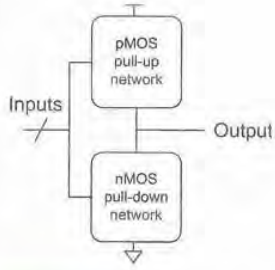


FIG 1.13 General logic gate using pull-up and pull-down networks

Table 1.3 Output states of CMOS logic gate

	pull-up OFF	pull-up ON
pull-down OFF	Z	1
pull-down ON	0	crowbarred (X)

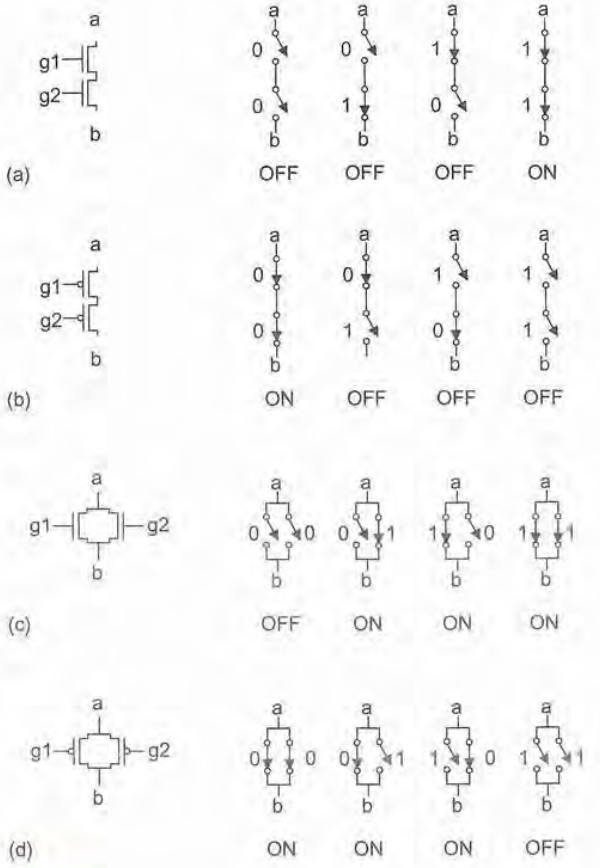


FIG 1.14 Connection and behavior of series and parallel transistors

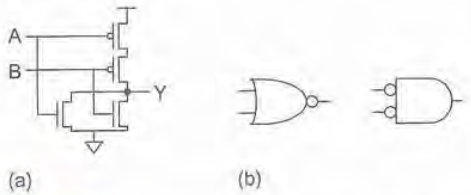


FIG 1.15 2-input NOR gate schematic (a) and symbol (b) $Y = \overline{A + B}$

1.4.4 The NOR Gate

A 2-input NOR gate is shown in Figure 1.15. The nMOS transistors are in parallel to pull the output low when either input is high. The pMOS transistors are in series to pull the output high when both inputs are low, as indicated by the truth table of Table 1.4. As with the NAND gate, there is never a case in which the output is crowbarred or left floating.

Table 1.4 NOR gate truth table

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

Example

Sketch a 3-input CMOS NOR gate.

Solution: Figure 1.16 shows such a gate. If any input is high, the output is pulled low through the parallel nMOS transistors. If all inputs are low, the output is pulled high through the series pMOS transistors.

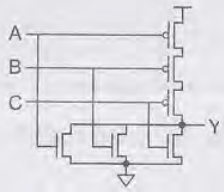


FIG 1.16 3-input NOR gate schematic
$Y = \overline{A + B + C}$

1.4.5 Compound Gates

A compound gate is formed by using a combination of series and parallel switch structures. For example, the derivation of the switch connection diagram for the function $Y = \overline{(A \cdot B) + (C \cdot D)}$ is shown in Figure 1.17. This function is sometimes called AND-OR-INVERT-22, or AOI22 because it performs the NOR of a pair of 2-input ANDs. For the nMOS pull-down network, take the uninverted expression $((A \cdot B) + (C \cdot D))$ indicating when the output should be pulled to '0.' The AND expressions $(A \cdot B)$ and $(C \cdot D)$ may be implemented by series connections of switches as shown in Figure 1.17(a). Now taking these as subswitches and ORing the result requires the parallel connection of these two structures, which is shown in Figure 1.17(b). For the pMOS pull-up network we must compute the complementary expression using switches that turn on with inversed polarity. By DeMorgan's Law, this is equivalent to interchanging AND and OR operations. Hence, transistors that appear in series in the pull-down network must appear in

parallel in the pull-up network. Transistors that appear in parallel in the pull-down network must appear in series in the pull-up network. This principle is called *conduction complements* and has already appeared in the design of the NAND and NOR gates. In the pull-up network, the parallel combination of A and B is placed in series with the parallel combination of C and D. This progression is evident in Figure 1.17(c) and Figure 1.17(d). Putting the networks together yields the connection diagram (Figure 1.17(e)). The schematic icon is shown in Figure 1.17(f), which shows that this gate can be used in a 2-input multiplexer. If $C = \overline{B}$, then $Y = A$ if B is true, while $Y = D$ if B is false.

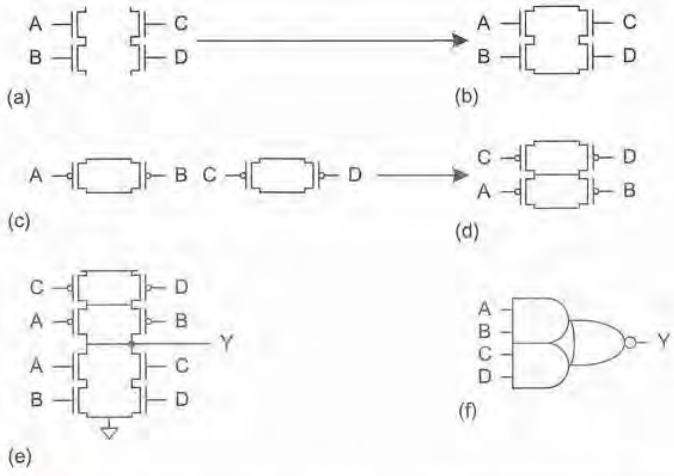


FIG 1.17 CMOS compound gate for function $Y = \overline{(A \bullet B) + (C \bullet D)}$

1.4.6 Pass Transistors and Transmission Gates

The *strength* of a signal is measured by how closely it approximates an ideal voltage source. In general, the stronger a signal, the more current it can source or sink. The power supplies, or *rails*, (V_{DD} and GND) are the source of the strongest '1's and '0's.

An nMOS transistor is an almost perfect switch when passing a '0' and thus we say it passes a *strong* '0.' However, the nMOS transistor is imperfect at passing a '1.' The high voltage level is somewhat less than V_{DD}, as will be explained in Section 2.3.2. We say it passes a *degraded* or *weak* '1.' A pMOS transistor again has the opposite behavior, passing strong '1's but degraded '0's. The transistor symbols and behaviors are summarized in Figure 1.19 with g, s, and d indicating gate, source, and drain. The symbols with arrows represent switches in the OFF and ON positions.

Example

Sketch a complementary CMOS gate computing $Y = \overline{(A + B + C) \cdot D}$.

Solution: Figure 1.18 shows such an OR-AND-INVERT-3-1 (OAI31) gate. The nMOS pull-down network pulls the output low if D is '1' and either A or B or C are '1', so D is in series with the parallel combination of A, B, and C. The pMOS pull-up network is the conduction complement, so D must be in parallel with the series combination of A, B, and C.

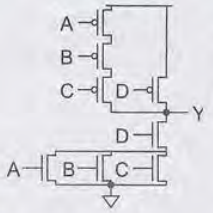


FIG 1.18 CMOS compound gate for function $Y = \overline{(A + B + C) \cdot D}$

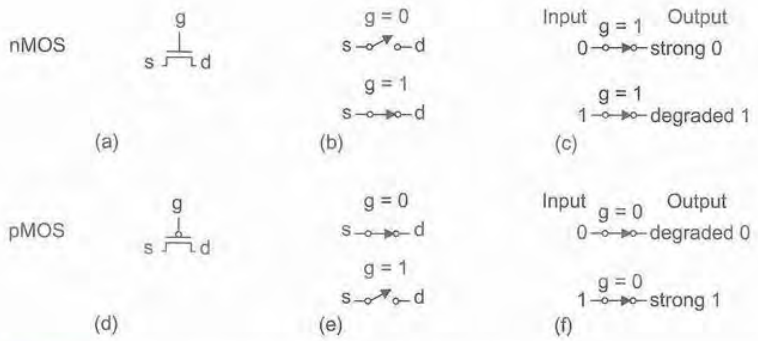


FIG 1.19 Pass transistor strong and degraded outputs

When an nMOS or pMOS is used alone as an imperfect switch, we sometimes call it a *pass transistor*. By combining an nMOS and a pMOS transistor in parallel (Figure 1.20(a)), we obtain a switch that turns on when a '1' is applied to g (Figure 1.20(b)) in which '0's and '1's are both passed in an acceptable fashion (Figure 1.20(c)). We term this a

transmission gate or *pass gate*. In a circuit where only a '0' or a '1' has to be passed, the appropriate transistor (n or p) can be deleted, reverting to a single nMOS or pMOS device. Note that both the control input and its complement are required by the transmission gate. This is called *double rail* logic. Some circuit symbols for the transmission gate are shown in Figure 1.20(d).[1] None are easier to draw than the simple schematic, so we will use the schematic to represent a transmission gate in this book.

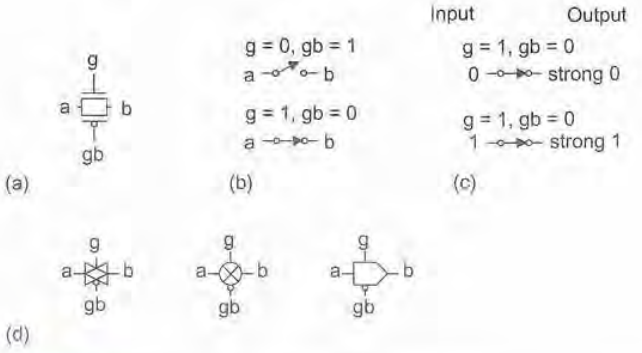


FIG 1.20 Transmission gate

In all of our examples so far, the inputs drive the gate terminals of nMOS transistors in the pull-down network and pMOS transistors in the complementary pull-up network, as was shown in Figure 1.13. Thus the nMOS transistors only need to pass 0's and the pMOS only pass 1's, so the output is always strongly driven and the levels are never degraded. This is called a *fully restored* logic gate and simplifies circuit design considerably. In contrast to other forms of logic, where the pull-up and pull-down switch networks have to be ratioed in some manner, complementary CMOS gates operate correctly independently of the physical sizes of the transistors. Moreover, there is never a path through 'ON' transistors from the '1' to the '0' supplies for any combination of inputs (in contrast to single-channel MOS, GaAs technologies, or bipolar). As we will learn in subsequent chapters, this is the basis for the low static power dissipation in CMOS.

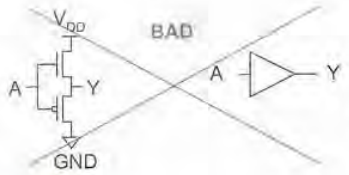


FIG 1.21 Bad noninverting buffer

A consequence of the design of complementary CMOS gates is that they must be inverting. The nMOS pull-down network turns ON when inputs are '1,' leading to '0' at the output. We might be tempted to turn the transistors upside down to build a noninverting gate. For example, Figure 1.21 shows a noninverting buffer. Unfortunately, now both the nMOS and pMOS transistors produce degraded outputs, so the technique should be avoided. Instead, we

[1] We call the left and right terminals *a* and *b* because each is technically the source of one of the transistors and the drain of the other.

can build noninverting functions from multiple stages of inverting gates. Figure 1.22 shows a 4-input AND gate built from two levels of inverting complementary CMOS gates. In isolation, the NAND design is simpler. In the context of a larger system, one can optimize the gates depending on the speed and density required.

Similarly, the compound gate of Figure 1.17 could be built with two AND gates, an OR gate, and an inverter. The AND and OR gates in turn could be constructed from NAND/NOR gates and inverters, shown in Figure 1.23, using a total of 20 transistors, as compared to 8 in Figure 1.17. CMOS logic designers must learn to take advantage of the efficiencies of compound gates rather than using large numbers of AND/OR gates.

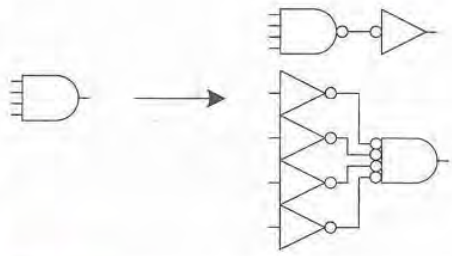


FIG 1.22 Various implementations of a CMOS 4-input AND gate

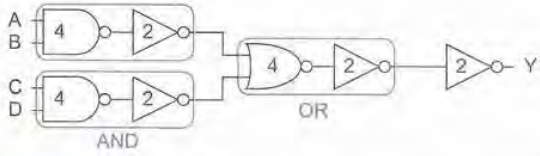


FIG 1.23 Inefficient discrete gate implementation of AOI22 indicating transistor counts

1.4.7 Tristates

Figure 1.24 shows symbols for a *tristate buffer*. When the enable input EN is '1,' the output Y equals the input A, just as in an ordinary buffer. When the enable is '0,' Y is left floating (a 'Z' value). This is summarized in Table 1.5. Sometimes both true and complementary enable signals EN and $\overline{\text{EN}}$ are drawn explicitly, while sometimes only EN is shown.



FIG 1.24 Tristate buffer symbol

Table 1.5 Truth table for tristate

EN / $\overline{\text{EN}}$	A	Y
0/1	0	Z
0/1	1	Z
1/0	0	0
1/0	1	1

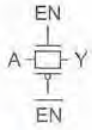


FIG 1.25 Transmission gate

The transmission gate in Figure 1.25 has the same truth table as a tristate buffer. It only requires two transistors but it is a *nonrestoring* circuit. If the input is a noisy or other-

wise degraded signal, the output will receive the same noise. After several stages of nonrestoring logic, a signal can become too degraded to recognize. We will see in Section 4.2 that the delay of a series of nonrestoring gates also increases quadratically with the number of gates in series.

Figure 1.26(a) shows a *tristate inverter*. The output is actively driven from V_{DD} or GND, so it is a restoring logic gate. Unlike any of the gates considered so far, the tristate inverter does not obey the conduction complements rule because it must allow the output to float under certain input combinations. When EN is '0' (Figure 1.26(b)), both enable transistors are OFF, leaving the output floating. When EN is '1' (Figure 1.26(c)), both enable transistors are ON. They are conceptually removed from the circuit, leaving a simple inverter. Figure 1.26(d) shows symbols for the tristate inverter. The complementary enable signal can be generated internally or can be routed to the cell explicitly. A tristate buffer can be built as a tristate inverter following an ordinary inverter.

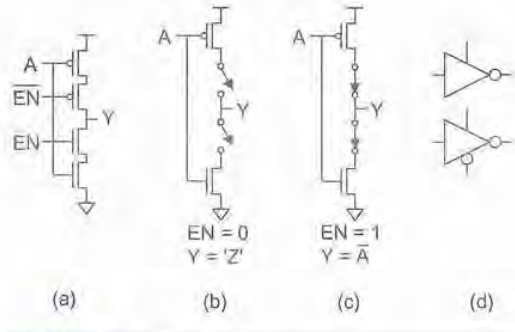


FIG 1.26 Tristate inverter

Tristates were once commonly used to allow multiple units to drive a common bus, as long as more than one are not simultaneously enabled. Distributing mutually exclusive enable signals in a timely fashion across a large chip is becoming more difficult, so multiplexers are now preferred.

1.4.8 Multiplexers

Multiplexers are key components in CMOS memory elements and data manipulation structures. A multiplexer chooses the output to be one of several inputs based on a select signal. A two-input, or 2:1 multiplexer, chooses input *D0* when the select is '0' and input *D1* when the select is '1.' The truth table is given in Table 1.6; the logic function is $Y = \overline{S} \cdot D0 + S \cdot D1$.

Table 1.6 Multiplexer truth table

$S / \overline{S}$	$D1$	$D0$	Y
0/1	X	0	0
0/1	X	1	1
1/0	0	X	0
1/0	1	X	1

Two transmission gates can be tied together to form a compact 2-input multiplexer, as shown in Figure 1.27(a). The select and its complement enable exactly one of the two transmission gates at any given time. The complementary select $\overline{S}$ is often not drawn in the symbol, as in Figure 1.27(b).

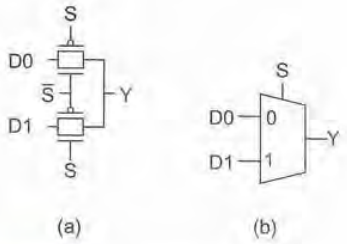


FIG 1.27 Transmission gate multiplexer

Again, the transmission gates produce a nonrestoring multiplexer. We could build a restoring, inverting multiplexer out of gates in several ways. One is the compound gate of Figure 1.17, connected as shown in Figure 1.28(a). Another is to gang together two tristate inverters, as shown in Figure 1.28(b). Notice that the schematics of these two approaches are nearly identical, save that the pull-up network has been slightly simplified and permuted in Figure 1.28(b). This is possible because the select and its complement are mutually exclusive. The tristate approach is slightly more compact and faster because it requires less internal wire. Again, if the complementary select is generated within the cell, it is omitted from the symbol (Figure 1.28(c)).

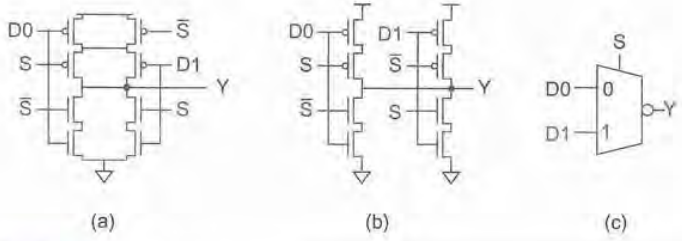


FIG 1.28 Inverting multiplexer

Larger multiplexers can be built from multiple 2-input multiplexers or by directly ganging together several tristates. The latter approach requires decoded select signals for each tristate. 4-input (4:1) multiplexers using each of these approaches are shown in Figure 1.29.

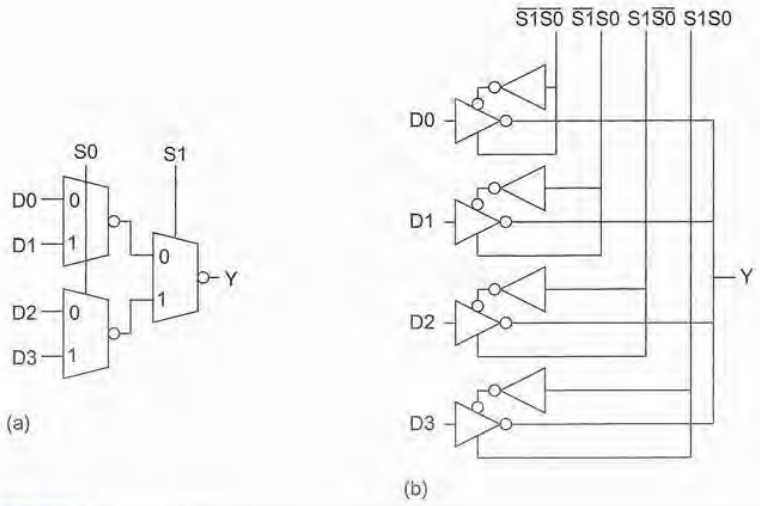


FIG 1.29 4:1 multiplexer

1.4.9 Latches and Flip-Flops

Using the combinational circuits developed so far, we can now build sequential circuits such as latches and flip-flops. A *D latch* using one 2-input multiplexer and two inverters is shown in Figure 1.30(a). It consists of a data input, D, a clock input, CLK, and true and complementary outputs Q and $\overline{Q}$. When CLK = '1,' the latch is *transparent*. $Q = D$ and $\overline{Q} = \overline{D}$ (Figure 1.30(c)). When CLK is switched to '0,' the latch is *opaque*. A feedback path around the inverter pair is established (Figure 1.30(d)) to hold the current state of Q indefinitely. While the latch is opaque, the input D is ignored. The multiplexer can be constructed from a pair of transmission gates, shown in Figure 1.30(b).

The D latch is also known as a *level-sensitive latch* because the state of the output is dependent on the level of the clock signal, as shown in Figure 1.30(e). The latch shown is a positive-level-sensitive latch, represented by the symbol in Figure 1.30(f). By inverting the control connections to the multiplexer, a negative-level-sensitive latch may be constructed.

By combining two level-sensitive latches, one positive-sensitive and one negative-sensitive, we construct an *edge-triggered flip-flop* as shown in Figure 1.31(a-b). By convention, the first latch stage is called the *master* and the second is called the *slave*.

While CLK is low, the master negative-level-sensitive latch output ($\overline{QM}$) follows the D input while the slave positive-level-sensitive latch holds the previous value (Figure 1.31(c)). When the clock transitions from 0 to 1, the master latch ceases to sample the input and holds the D value at the time of the clock transition. The slave latch opens, passing the stored master value ($\overline{QM}$) to the output of the slave latch (Q). The D input is blocked from affecting the output because the master is disconnected from the D input

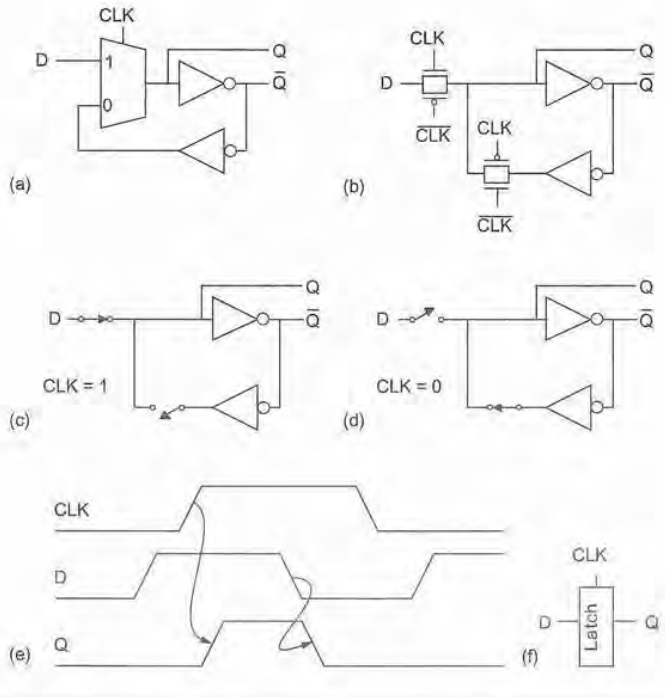


FIG 1.30 CMOS positive-level-sensitive D latch

(Figure 1.31(d)). When the clock transitions from 1 to 0, the slave latch holds its value and the master starts sampling the input again.

In summary, this flip-flop copies D to Q on the rising edge of the clock, as shown in Figure 1.31(e). Thus this device is called a positive-edge triggered flip-flop (also called a *D flip-flop*, *D register*, or *master–slave flip-flop*). Figure 1.31(f) shows the circuit symbol for the flip-flop. By reversing the latch polarities, a negative edge triggered flip-flop may be constructed. A collection of two or more D flip-flops sharing a common clock input is called a *register*. A register is often drawn as a flip-flop with multi-bit D and Q busses.

In Section 7.2.3 we will see that flip-flops may experience hold-time failures if the system has too much *clock skew*, i.e., if one flip-flop triggers early and another triggers late because of variations in clock arrival times. In industrial designs, a great deal of effort is devoted to timing simulations to catch hold-time problems. When design time is more important (e.g., in academic class projects), hold time problems can be avoided altogether by distributing a two-phase nonoverlapping clock. Figure 1.32 shows the flip-flop clocked with two nonoverlapping phases. As long as the phases do not overlap even with worst-case skews, at least one latch will be opaque at any given time and hold-time problems will never occur.

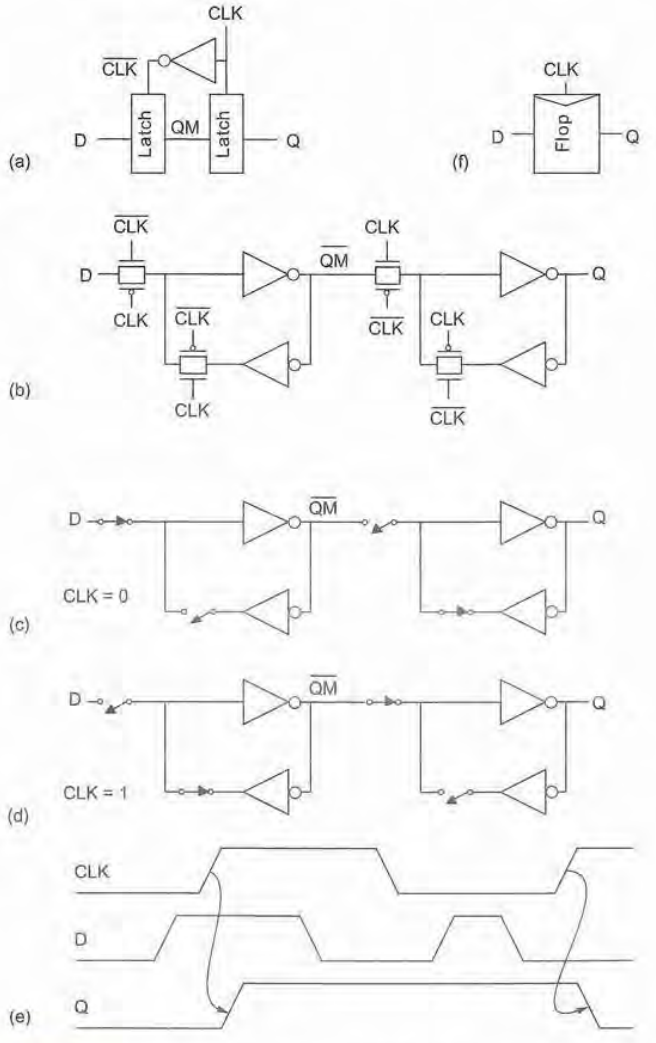


FIG 1.31 CMOS positive-edge-triggered D flip-flop

It is often useful to provide reset and/or enable signals to flip-flops and latches. Such modifications are straightforward and are discussed in Section 7.3.

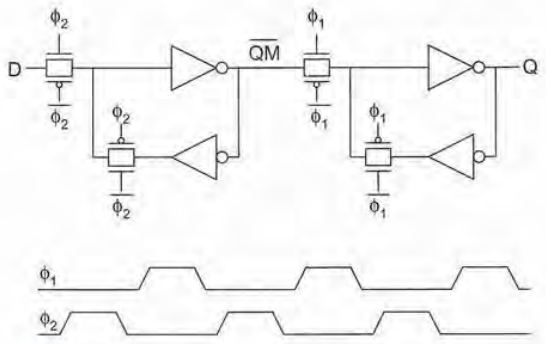


FIG 1.32 CMOS flip-flop with two-phase nonoverlapping clocks

1.5 CMOS Fabrication and Layout

Now that we can design logic gates and latches from transistors, let us consider how the transistors are built. Transistors are fabricated on thin silicon wafers that serve as both a mechanical support and an electrical common point called the *substrate*. We can understand the physical layout of transistors from two perspectives. One is the top view, obtained by looking down on a wafer. The other is the cross-section, obtained by slicing the wafer through the middle of a transistor and looking at it edgewise. We begin by looking at the cross-section of a complete CMOS inverter. We then look at the top view of the same inverter and define a set of masks used to manufacture the different parts of the inverter. The size of the transistors and wires is set by the mask dimensions and is limited by the resolution of the manufacturing process. Continual advancements in this resolution have fueled the exponential growth of the semiconductor industry.

1.5.1 Inverter Cross-section

Figure 1.33 shows a cross-section of the inverter from Section 1.4.1. In this diagram, the inverter is built on a p-type substrate. The pMOS transistor requires an n-type body region, so an n-well is diffused into the substrate in its vicinity. Note that it is also possible to design a CMOS process with an n-type substrate and p-wells to contain the nMOS transistors. As described in Section 1.3, the nMOS transistor has n-type source and drain regions and a polysilicon gate over a thin layer of silicon dioxide (SiO_2, also called *gate oxide*). The pMOS transistor is a similar structure with p-type source and drain regions. The polysilicon gates of the two transistors are tied together somewhere off the page and form the input A. The source of the nMOS transistor is connected to a metal ground line and the source of the pMOS transistor is connected to a metal V_{DD} line. The drains of the

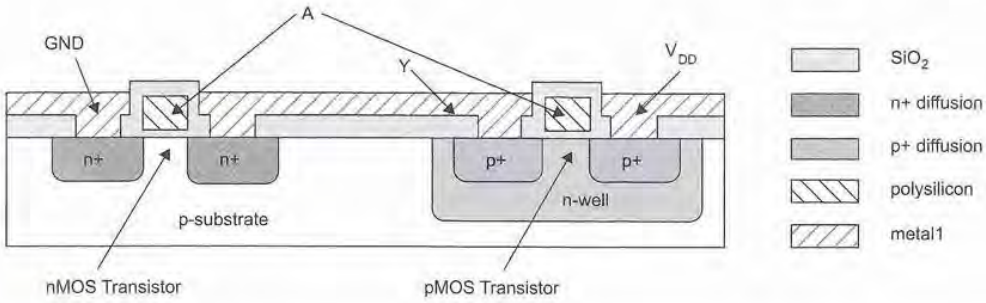


FIG 1.33 Inverter cross-section

two transistors are connected with metal to form the output *Y*. A thicker layer of SiO_2 called *field oxide* prevents metal from shorting to other layers except where contacts are explicitly etched.

The substrate must be tied to a low potential to avoid forward-biasing the p-n junction between the p-type substrate and the n+ nMOS source or drain. Likewise, the n-well must be tied to a high potential. This is generally done by adding heavily doped substrate and well contacts, or *taps*, to connect GND and V_{DD} to the substrate and n-well, respectively, as shown in Figure 1.34 and the inside front cover. The heavy doping is required to establish a good *ohmic contact* that provides low resistance for bidirectional current flow; a metal to lightly doped semiconductor junction forms a *Schottky diode*.

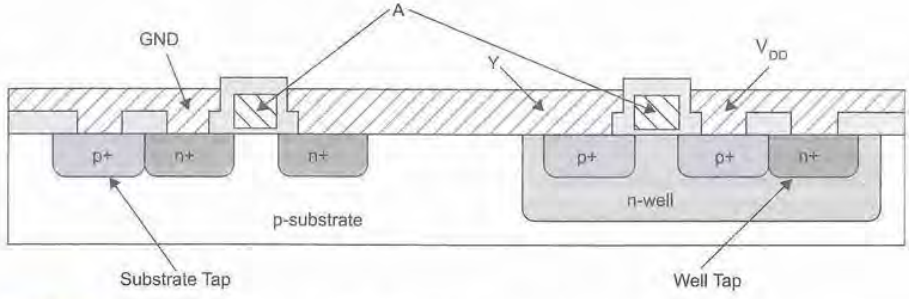


FIG 1.34 Inverter cross-section with well and substrate contacts. Color version on inside front cover.

1.5.2 Fabrication Process

For all their complexity, chips are amazingly inexpensive because all the transistors and wires can be printed in much the same way as books. The fabrication sequence consists of a

series of steps in which layers of the chip are defined through a process called *photolithography*. Because many entire chips are printed at once, the cost of the chip is proportional to the chip area, rather than the number of transistors. As manufacturing advances allow engineers to build smaller transistors and place more transistors in the same area, each transistor gets cheaper. Smaller transistors are also faster because electrons don't have to travel as far to get from the source to the drain! This explains the remarkable trend for computers and electronics to become both cheaper and more capable with each generation.

The inverter could be defined by a hypothetical set of six masks: n-well, polysilicon, n+ diffusion, p+ diffusion, contacts, and metal (for fabrication reasons discussed in Chapter 3, the actual mask set is usually different). Masks specify where the components will be manufactured on the chip. Figure 1.35(a) shows a top view of the six masks. (See also the inside front cover for a color picture.) The cross-section of the inverter from Figure 1.34 was taken along the dashed line.

Consider a very simple fabrication process to illustrate the fundamental ideas. The process begins with the creation of an n-well on a bare p-type silicon wafer. Figure 1.36 shows cross-sections of the wafer after each processing step involved in forming the n-well; Figure 1.36(a) illustrates the bare substrate before processing. Forming the n-well requires adding enough Group V dopants into the silicon substrate to change the substrate from p-type to n-type in the region of the well. To define what regions receive n-wells, we grow a protective layer of oxide over the entire wafer, then remove it where we want the wells. We then add the n-type dopants; the dopants are blocked by the oxide, but enter the substrate and form the wells where there is no oxide. The next paragraph describes these steps in more detail.

The wafer is first *oxidized* in a high-temperature (typically 900°–1200°C) furnace that causes the Si and O_2 to react and become SiO_2 on the wafer surface (Figure 1.36(b)). The oxide must be patterned to define the n-well. An organic photoresist[2] that softens where exposed to light is spun onto the wafer (Figure 1.36(c)). The photoresist is exposed through the n-well mask (Figure 1.35(b)) that allows light to pass through only where the well should be. The softened photoresist is removed to expose the oxide (Figure 1.36(d)). The oxide is etched with hydrofluoric acid (HF) where it is not protected by the photoresist (Figure 1.36(e)), then the remaining photoresist is stripped away using a mixture of acids called *piranha etch* (Figure 1.36(f)). The well is formed where the substrate is not covered with oxide. Two ways to add dopants are diffusion and ion implantation. In the *diffusion* process, the wafer is placed in a furnace with a gas containing the dopants. When heated, dopant atoms diffuse into the substrate. Notice how the well is larger than the hole in the oxide on account of *lateral* diffusion (Figure 1.36(g)). With *ion implantation*, dopant ions are accelerated through an electric field and blasted into the substrate. In either method, the oxide layer prevents dopant atoms from entering the substrate where no well is intended. Finally, the remaining oxide is stripped with HF to leave the bare wafer with wells in the appropriate places.

[2]Engineers have experimented with many organic polymers. Brumford and Walker reported in 1958 that Jello™ could be used for masking. They did extensive testing, observing that "various Jellos™ were evaluated with lemon giving the best result."

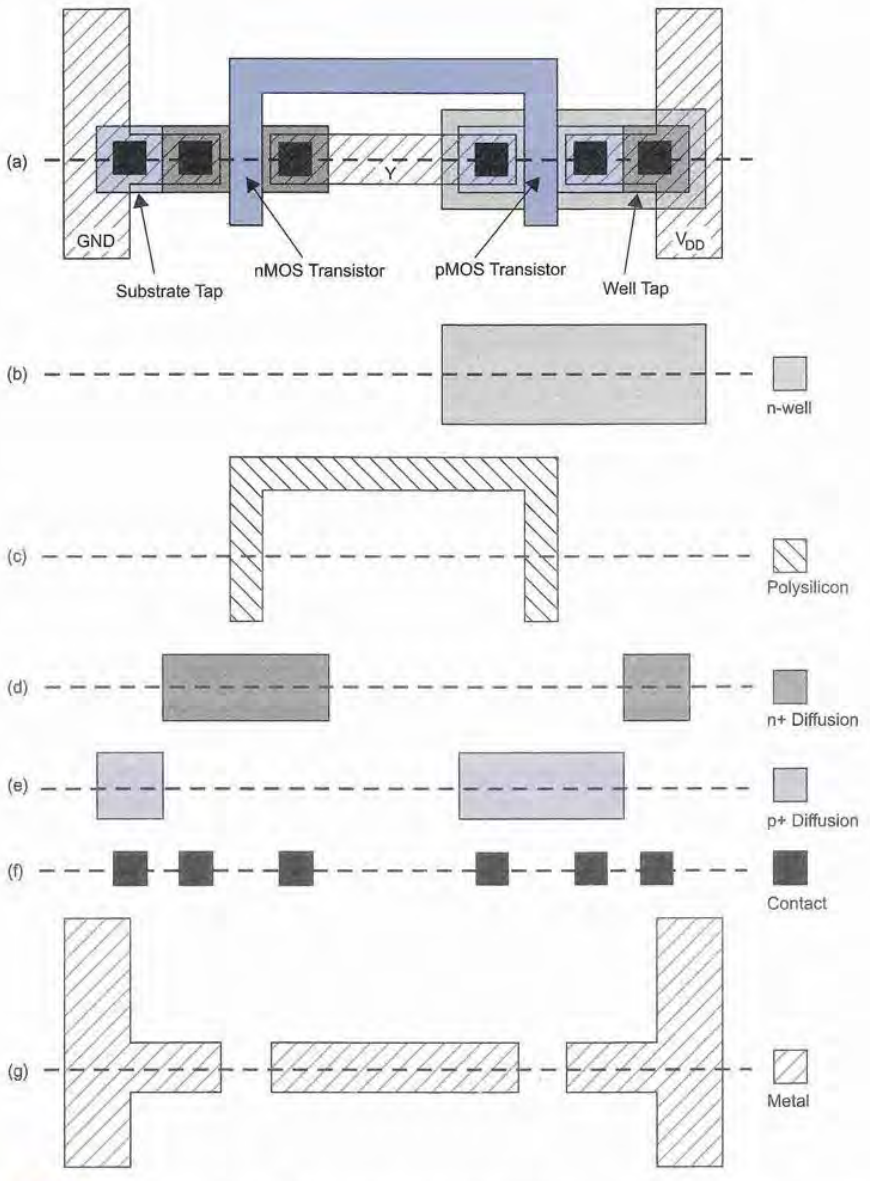


FIG 1.35 Inverter mask set. Color version on inside front cover.

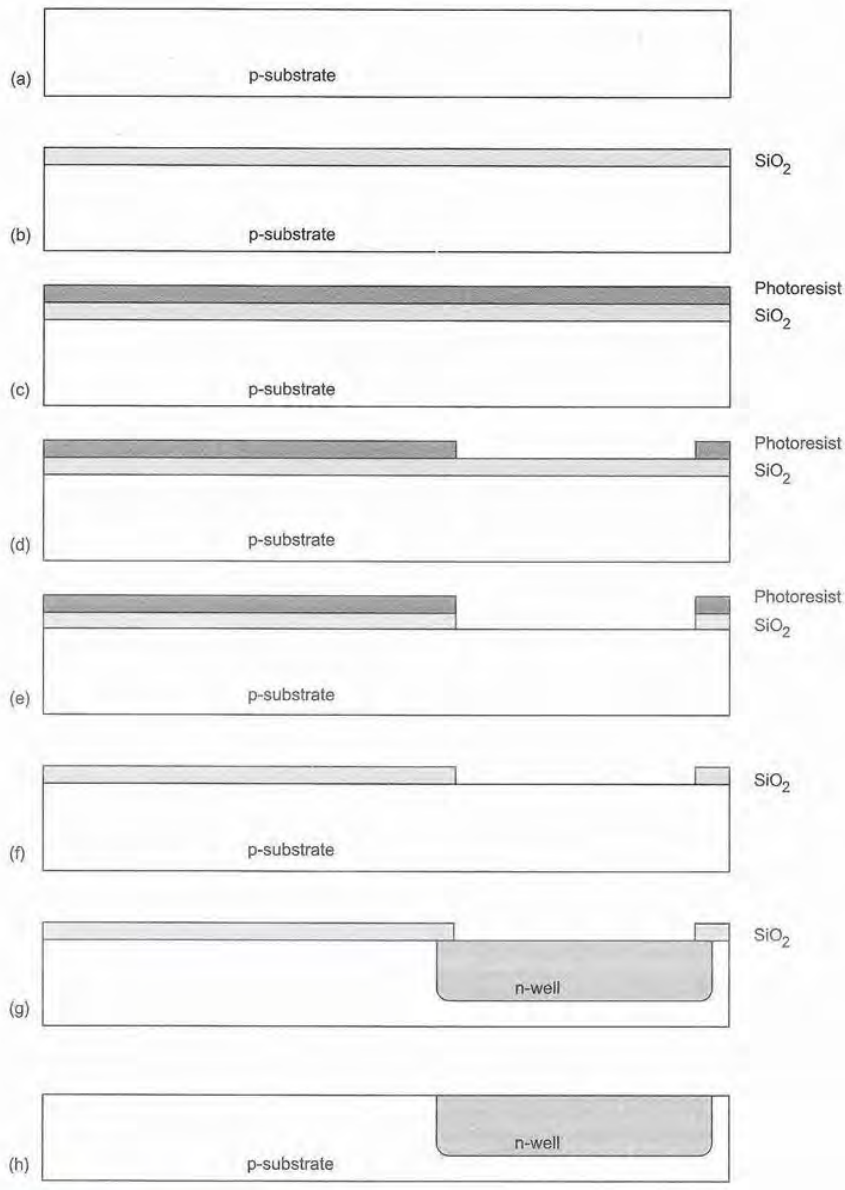


FIG 1.36 Cross-sections while manufacturing the n-well

The transistor gates are formed next. These consist of polycrystalline silicon, generally called *polysilicon*, over a thin layer of oxide. The thin oxide is grown in a furnace. Then the wafer is placed in a reactor with silane gas (SiH_4) and heated again to grow the polysilicon layer through a process called *chemical vapor deposition*. The polysilicon is heavily doped to form a reasonably good conductor. The resulting cross-section is shown in Figure 1.37(a). As before, the wafer is patterned with photoresist and the polysilicon mask (Figure 1.35(c)), leaving the polysilicon gates (Figure 1.37(b)).

The n+ regions are diffused to create the transistor active area and the well contact. As with the well, a protective layer of oxide is formed (Figure 1.37(c)) and patterned with the n-diffusion mask (Figure 1.35(d)) to expose the areas where the dopants are needed (Figure 1.37(d)). Although the n+ regions are typically formed with ion implantation (Figure 1.37(e)), they were historically diffused and thus still are often called *n-diffusion*. Notice that the polysilicon gate over the nMOS transistor blocks the diffusion so the source and drain are separated by a channel under the gate. This is called a *self-aligned* process because the source and drain of the transistor are automatically formed adjacent to the gate without the need to precisely align the masks. Finally, the protective oxide is stripped (Figure 1.37(f)).

The process is repeated for the p-diffusion mask (Figure 1.35(e)) to give the structure of Figure 1.38(a). Oxide is used for masking in the same way, and thus is not shown. The field oxide is grown to insulate the wafer from metal and patterned with the contact mask (Figure 1.35(f)) to leave contact cuts where metal should attach to diffusion or polysilicon (Figure 1.38(b)). Finally, aluminum is sputtered over the entire wafer, filling the contact cuts as well. Sputtering involves blasting aluminum into a vapor that evenly coats the wafer. The metal is patterned with the metal mask (Figure 1.35(g)) and plasma etched to remove metal everywhere except where wires should remain (Figure 1.38(c)). This completes the simple fabrication process.

Modern fabrication sequences are somewhat more elaborate because they must create complex doping profiles around the channel of the transistor and print features that are smaller than the wavelength of the light being used in lithography. However, masks for these elaborations can be automatically generated from the simple set of masks we have just examined. Modern processes may also have five or more layers of metal, so the metal and contact steps must be repeated for each layer. Chip manufacturing has become a commodity, and many different vendors will build designs from a basic set of masks.

1.5.3 Layout Design Rules

Layout design rules describe how small features can be and how closely they can be packed in a particular manufacturing process. Industrial design rules are usually specified in microns. This makes migrating from one process to a more advanced process difficult because not all rules scale in the same way. Mead and Conway [Mead80] popularized lambda-based design rules based on a single parameter, λ, which characterizes the resolution of the process. λ is generally half of the minimum drawn transistor channel length.

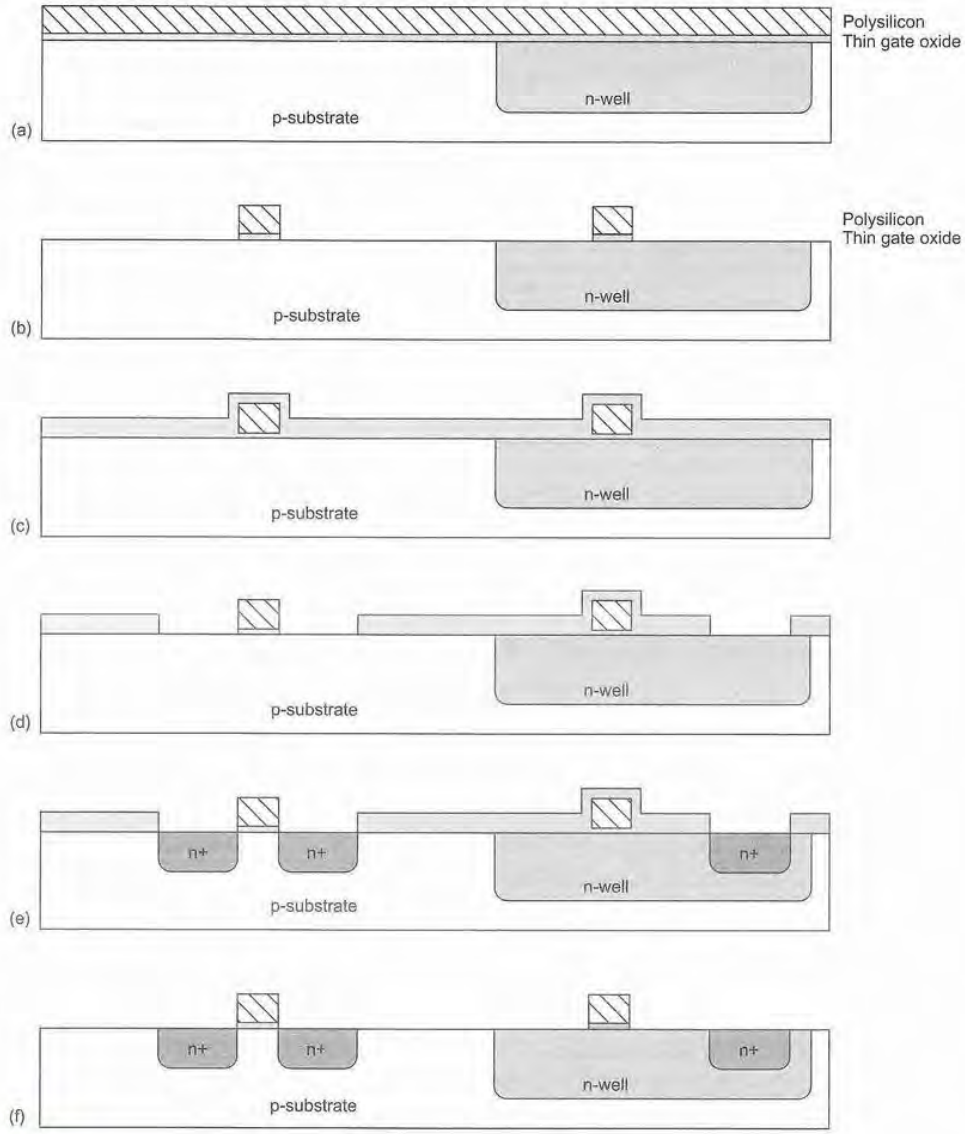


FIG 1.37 Cross-sections while manufacturing polysilicon and n-diffusion

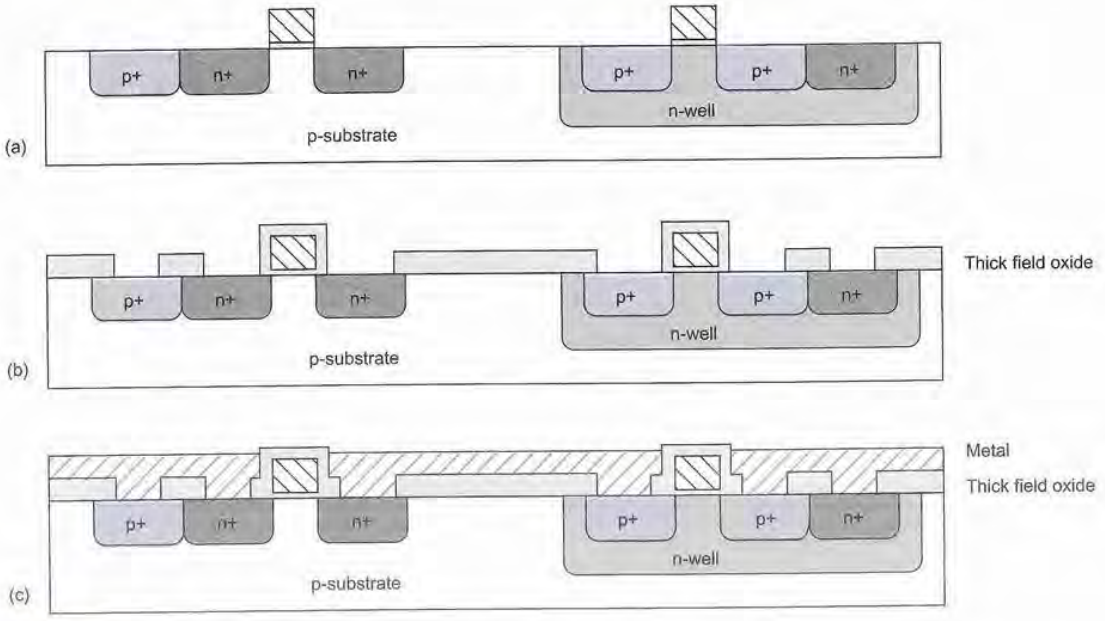


FIG 1.38 Cross-sections while manufacturing p-diffusion, contacts, and metal

This length is the distance between the source and drain of a transistor and is set by the minimum width of a polysilicon wire. This dimension is typically specified in microns for dimensions above 0.18 μm = 180 nm and in nanometers below. For example, a 180 nm process has a minimum polysilicon width (and hence transistor length) of 0.18 microns and uses design rules with λ = 0.09 μm[3]. Lambda-based rules are necessarily conservative because they round dimensions up to an integer multiple of λ. However, they make scaling layout trivial; the same layout can be moved to a new process simply by specifying a new value of λ. This chapter will present design rules in terms of λ. The potential density advantage of micron rules is sacrificed for simplicity and easy scalability of lambda rules. Designers often describe a process by its *feature size*. Feature size refers to minimum transistor length, so λ is half the feature size.

The MOSIS service [Piña02] is a low-cost prototyping service that collects designs from academic, commercial, and government customers and aggregates them onto one mask set to share overhead costs and generate production volumes sufficient to interest fabrication companies. MOSIS has developed a set of scalable lambda-based design rules

[3]Some 180 nm lambda-based rules actually set λ = 0.10 μm, then shrink the gate by 20 nm while generating masks. This keeps 180 nm gate lengths but makes all other features slightly larger.

that cover a wide range of manufacturing processes. The rules have become slightly more conservative for more advanced submicron processes, but are fundamentally very similar to those proposed by Mead and Conway in 1980 for a 6-micron process. The rules describe the minimum width to avoid breaks in a line, minimum spacing to avoid shorts between lines, and minimum overlap to ensure two layers completely overlap.

A conservative but easy-to-use set of design rules for layouts with two metal layers in an n-well process is as follows.

- Metal and diffusion have minimum width and spacing of 4 λ.
- Contacts are 2 λ × 2 λ and must be surrounded by 1 λ on the layers above and below.
- Polysilicon uses a width of 2 λ.
- Polysilicon overlaps diffusion by 2 λ where a transistor is desired and has a spacing of 1 λ away where no transistor is desired.
- Polysilicon and contacts have a spacing of 3 λ from other polysilicon or contacts.
- N-well surrounds pMOS transistors by 6 λ and avoids nMOS transistors by 6 λ.

Figure 1.39 shows the basic MOSIS design rules for a process with two metal layers. Section 3.3 elaborates on these rules and compares them with more aggressive industrial design rules.

Transistor dimensions are often specified by their Width/Length (W/L) ratio. For example, the nMOS transistor in Figure 1.39 formed where polysilicon crosses n-diffusion

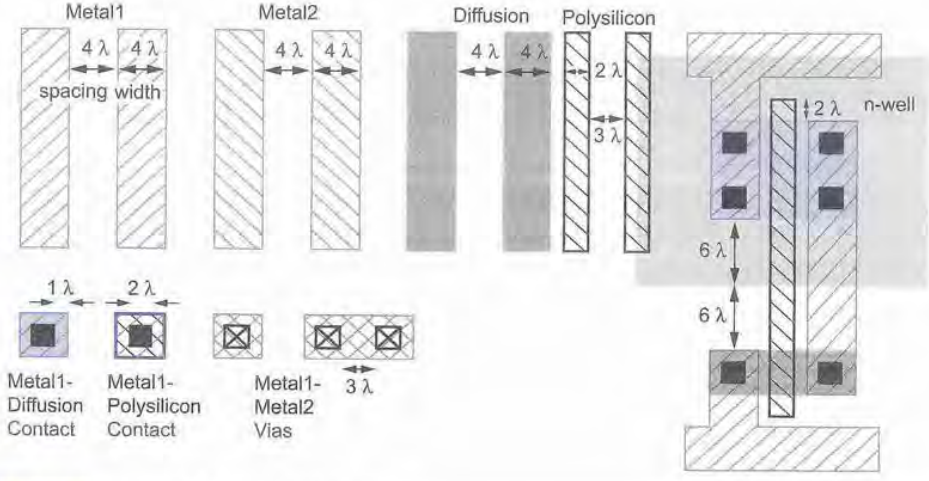


FIG 1.39 Simplified λ-based design rules

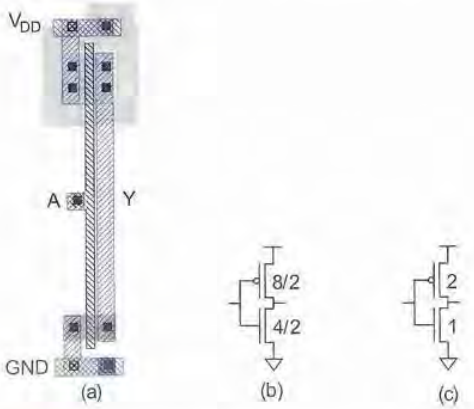


FIG 1.40 Inverter with dimensions labeled

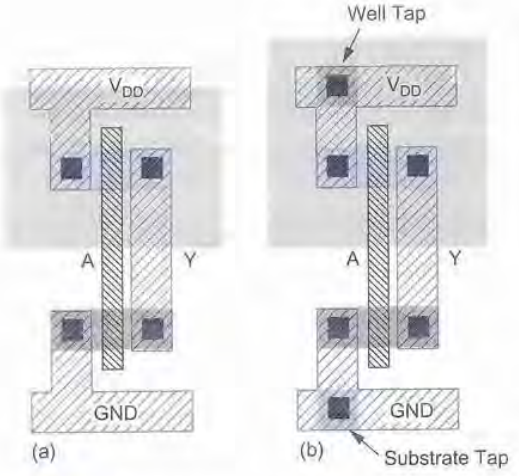


FIG 1.41 Inverter standard cell layout

has a W/L of 4/2. In a 0.6 μm process, this corresponds to an actual width of 1.2 μm and a length of 0.6 μm. Such a minimum-width contacted transistor is often called a unit transistor.[4] pMOS transistors are often wider than nMOS transistors because holes move more slowly than electrons so the transistor has to be wider to deliver the same current. Figure 1.40(a) shows a unit inverter layout with a unit nMOS transistor and a double-sized pMOS transistor. Figure 1.40(b) shows a schematic for the inverter annotated with Width/Length for each transistor. Figure 1.40(c) shows a shorthand we will often use, specifying multiples of unit width and assuming minimum length.

1.5.4 Gate Layout

A good deal of ingenuity can be exercised and a vast amount of time wasted exploring layout topologies to minimize the size of a gate or other *cell* such as an adder or memory element. For many applications, a straightforward layout is good enough and can be automatically generated or rapidly built by hand. This section presents a simple layout style based on a "line of diffusion" rule that is commonly used for standard cells in automated layout systems. This style consists of four horizontal strips: metal ground at the bottom of the cell, n-diffusion, p-diffusion, and metal power at the top. The power and ground lines are often called *supply rails*. Polysilicon lines run vertically to form transistor gates. Metal wires within the cell connect the transistors appropriately.

Figure 1.41(a) shows such a layout for an inverter. The input *A* can be connected from the top, bottom, or left in polysilicon. The output *Y* is available at the right side of the cell in metal. Recall that the p-substrate and n-well must be tied to ground and power, respectively. Figure 1.41(b) shows the same inverter with well and substrate taps placed under the power and ground rails, respectively. Figure 1.42 shows a 3-input NAND gate. Notice how the nMOS transistors are connected in series while the pMOS transistors are connected in parallel. Power and ground extend 2 λ on each side so if two gates were abutted the contents would be separated by 4 λ, satisfying design rules. The height of the cell is 36 λ, or 40 λ if the 4 λ space between the cell and another wire above it is counted. All these examples use transistors of width 4 λ. Choice of transistor width is addressed further in Chapter 4.

[4]Such small transistors in modern processes often behave slightly differently than their wider counterparts. Moreover, the transistor will not operate if either contact is damaged. Industrial designers often use a transistor wide enough for two contacts (9 λ) as the unit transistor to avoid these problems.

These cells were designed such that the gate connections are made from the top or bottom in polysilicon. In contemporary standard cells, polysilicon is generally not used as a routing layer so the cell must allow metal2 to metal1 and metal1 to polysilicon contacts to each gate. While this increases the size of the cell, it allows free access to all terminals on metal routing layers.

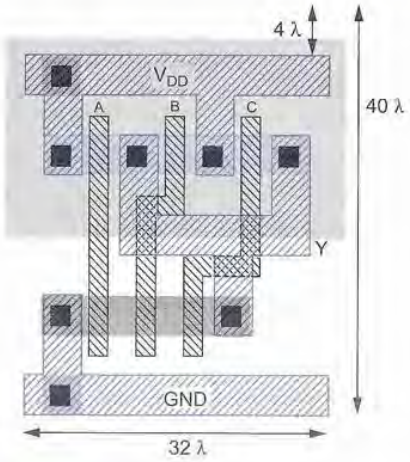


FIG 1.42 3-input NAND standard cell gate layouts

1.5.5 Stick Diagrams

As layout is time-consuming, it is important to have fast ways to plan layout and estimate area before committing to a full layout. *Stick diagrams* are easy to draw because they do not need to be drawn to scale. Figure 1.43 and the inside front cover show stick diagrams for an inverter and a 3-input NAND gate. While this book uses stipple patterns, layout designers use colored pencils or dry-erase markers.

With practice it is easy to estimate the area of a layout from the corresponding stick diagram even though the diagram is not to scale. Although schematics focus on transistors, layout area is usually determined by the metal wires. Transistors are merely small widgets that fit under the wires. We define a *routing track* as enough space to place a wire and the required spacing to the next wire. If our wires have a width of 4 λ and a spacing of 4 λ to the next wire, the track *pitch* is 8 λ, as shown in Figure 1.44(a). This pitch also leaves room for a transistor to be placed between the wires (Figure 1.44(b)). Therefore, as a rule of thumb, it is reasonable to estimate the height and width of a cell by counting the number of metal tracks and multiplying by 8 λ. A slight complication is the required spacing of 12 λ between nMOS and pMOS transistors set by the well, as shown in Figure 1.45(a). This space can be occupied by an additional track of wire, shown in Figure 1.45(b). Therefore, an extra track must be allocated between nMOS and pMOS transistors regardless of whether wire is actually used in that track. Figure 1.46 shows how to count tracks to estimate the size of a 3-input NAND. There are four vertical wire tracks, multiplied by 8 λ per track to give a cell width of 32 λ. There are five horizontal tracks, giving a cell height

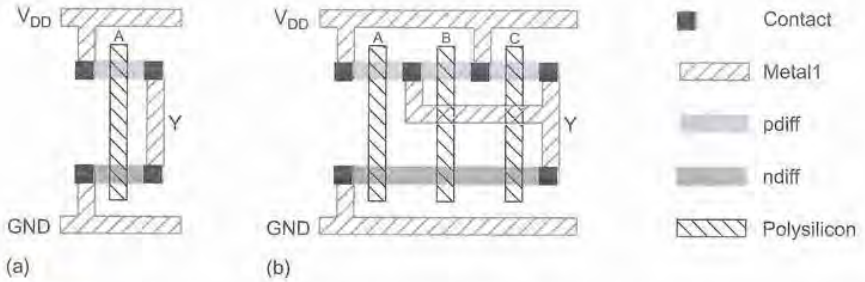


FIG 1.43 Stick diagrams of inverter and 3-input NAND gate. Color version on inside front cover.

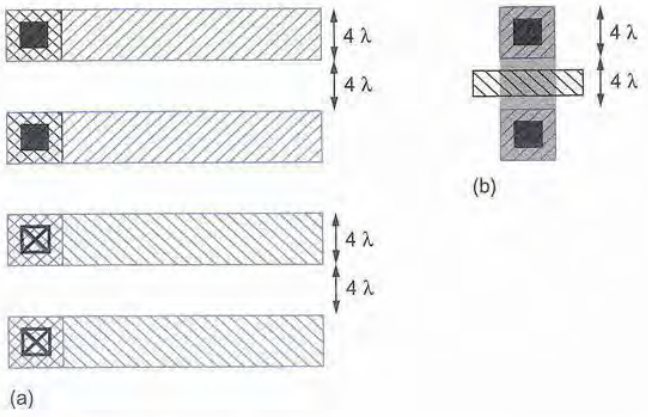


FIG 1.44 Pitch of routing tracks

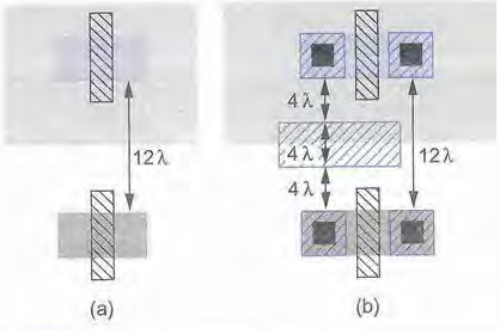


FIG 1.45 Spacing between nMOS and pMOS transistors

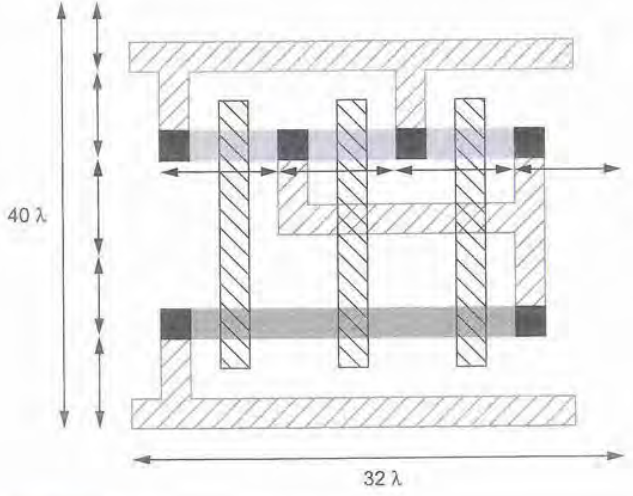


FIG 1.46 3-input NAND gate area estimation

of 40 λ. Even though the horizontal tracks are not drawn to scale, they are still easy to count. Figure 1.42 shows that the actual NAND gate layout agrees in size if the 4 λ spacing between the top of V_{DD} and the next place a metal line could be placed is counted. If transistors are wider than 4 λ, the extra width must be factored into the area estimate. Of course, these estimates are oversimplifications of the complete design rules and a trial layout should be performed for truly critical cells.

Example

Sketch a stick diagram for a CMOS gate computing $Y = \overline{(A + B + C) \cdot D}$ (see Figure 1.18) and estimate the cell width and height.

Solution: Figure 1.47 shows a stick diagram. Counting horizontal and vertical pitches gives an estimated cell size of 40 by 48 λ.

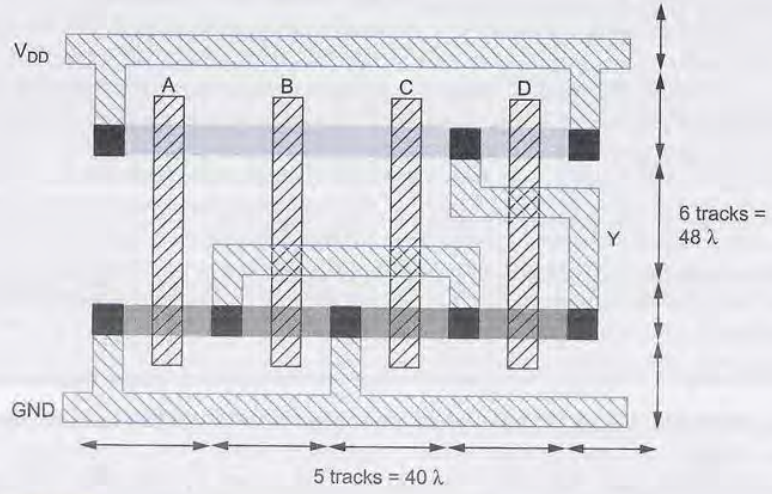


FIG 1.47 CMOS compound gate for function $Y = \overline{(A + B + C) \cdot D}$

1.6 Design Partitioning

By this point, you know that MOS transistors behave as voltage-controlled switches. You know how to build logic gates out of transistors. And you know how transistors are fabricated and how to draw a layout that specifies how transistors should be placed and connected together. You know enough to start building your own simple chips.

The greatest challenge in modern VLSI design is not in designing the individual transistors but rather in managing system complexity. Modern *System-On-Chip* (SOC) designs combine memories, processors, high speed I/O interfaces, and dedicated application-specific logic on a single chip. They use hundreds of millions (soon billions) of transistors. The implementation must be divided among large teams of engineers and each engineer must be highly productive. If the implementation is too rigidly partitioned, each block can be optimized without regard to its neighbors, leading to poor system results. Conversely, if every task is interdependent with every other task, design will progress too

slowly. Design managers face the challenge of choosing a suitable tradeoff between these extremes. There is no substitute for practical experience in making these choices, and talented engineers who have experience with multiple designs are very important to the success of a large project. The notion of structured design, which is also used in large software projects, will be introduced in Chapter 8. Structured design uses the principles of hierarchy, regularity, modularity, and locality to manage the complexity.

Digital VLSI design is often partitioned into five interrelated tasks: *architecture* design, *microarchitecture* design, *logic* design, *circuit* design, and *physical* design. Architecture describes the functions of the system. For example, the x86 microprocessor architecture specifies the instruction set, register set, and memory model. Microarchitecture describes how the architecture is partitioned into registers and functional units. The 80386, 80486, Pentium, Pentium II, Pentium III, Pentium 4, Celeron, Cyrix MII, AMD K5, and Athlon are all microarchitectures offering different performance / transistor count tradeoffs for the x86 architecture. Logic describes how functional units are constructed. For example, various logic designs for a 32-bit adder in the x86 integer unit include ripple carry, carry lookahead, and carry select. Circuit design describes how transistors are used to implement the logic. For example, a carry lookahead adder can use static CMOS circuits, domino circuits, or pass transistors. The circuits can be tailored to emphasize high performance or low power. Physical design describes the layout of the chip.

These elements are inherently interdependent. For example, choices of microarchitecture and logic are strongly dependent on the number of transistors that can be placed on the chip, which depends on the physical design and process technology. Similarly, innovative circuit design that reduces a cache access from two cycles to one can influence which microarchitecture is most desirable. The choice of clock frequency depends on a complex interplay of microarchitecture and logic, circuit design, and physical design. Deeper pipelines allow higher frequencies but lead to greater performance penalties when operations early in the pipeline are dependent on those late in the pipeline. Many functions have various logic and circuit designs trading speed for area, power, and design effort. Custom physical design allows more compact, faster circuits and lower manufacturing costs, but involves an enormous labor cost. Automatic layout with CAD systems reduces the labor and achieves faster times to market.

To deal with these interdependencies, microarchitecture, logic, circuit, and physical design must occur, at least in part, in parallel. Microarchitects depend on circuit and physical design studies to understand the cost of proposed microarchitectural features. Engineers are sometimes categorized as "short and fat" or "tall and skinny." Tall, skinny engineers understand something about a broad range of topics. Short, fat engineers understand a large amount about a narrow field. Digital VLSI design favors the tall, skinny engineer who can evaluate how choices in one part of the system impact other parts of the system.

A critical tool for managing complex designs is *hierarchy*. A large system can be partitioned into many *units*. Each unit in turn is composed of multiple *functional blocks*[5]. These blocks in turn are built from *cells*, which ultimately are constructed from transistors. The

[5]Some designers refer to both units and functional blocks as *modules*.

system can be more easily understood at the top level by viewing units as black boxes with well-defined interfaces and functions rather than looking at each individual transistor. Hierarchy also facilitates design reuse; a block can be designed and verified once, then used in many places. Logic, circuit, and physical views of the design should share the same hierarchy for ease of verification. A design hierarchy can be viewed as a tree structure with the overall chip as the *root* and the primitive cells as *leafs*.

An alternative way of viewing design partitioning is shown with the Y-chart in Figure 1.48 [Gajski83, Kang03]. The radial lines on the Y-chart represent three distinct design domains: behavioral, structural, and physical. These domains can be used to describe the design of almost any artifact and thus form a very general taxonomy for describing the design process. Within each domain there are a number of levels of design abstraction that start at a very high level and descend eventually to the individual elements that need to be aggregated to yield the top level function (i.e., in the case of chip design and transistors).

The behavioral domain describes what a particular system does. For instance, at the highest level we might state that we desire to build a chip that can generate audio tones of specific frequencies (i.e., a touch-tone generator for a telephone). This behavior can be successively refined to more precisely describe what needs to be done in order to build the tone generator (i.e., the frequencies desired, output levels, distortion allowed, etc.).

At each abstraction level, a corresponding structural description can be described. The structural domain describes the interconnection of modules necessary to achieve a particular behavior. For instance, at the highest level, the touch-tone generator might consist of a keyboard, a tone generator, an audio amplifier, a battery, and a speaker. Eventually at lower levels of abstraction, the individual gate and then transistor connections required to build the tone generator are described.

For each level of abstraction, the physical domain description explains how to physically construct that level of abstraction. At high levels this might consist of an engineering drawing showing how to put together the keyboard, tone generator chip, battery, and speaker in the associated housing. At the top chip level, this might consist of a floorplan, and at lower levels, the actual geometry of individual transistors.

The design process can be viewed as making transformations from one domain to another while maintaining the equivalency of the domains. Behavioral descriptions are transformed to structural descriptions, which in turn are transformed to physical descriptions. These transformations can be manual or automatic. In either case, it is normal design practice to verify the transformation of one domain to the other by some checking process. This ensures that the design intent is carried across the domain boundaries. Hierarchically specifying each domain at successively detailed levels of abstraction allows us to design very large systems.

The reason for strictly describing the domains and levels of abstraction is to define a precise design process in which the final function of the system can be traced all the way back to the initial behavioral description. There should be no opportunity to produce an incorrect design. If anomalies arise, the design process is corrected so that those anomalies will not reoccur in the future. A designer should acquire a rigid discipline with respect to the design process, and be aware of each transformation and how and why it is failproof.

Normally, these steps are fully automated in a modern design process, but it is important to be aware of the basis for these steps in order to debug them if they go astray.

The Y diagram can be used to illustrate each domain and the transformations between domains at varying levels of design abstraction. As the design process winds its way from the outer to inner rings, it proceeds from higher to lower levels of abstraction and hierarchy.

Most of the remainder of this chapter is a case study in the design of a simple microprocessor to illustrate the various aspects of VLSI design applied to a nontrivial system. We begin by describing the architecture and microarchitecture of the processor. We then consider logic design and discuss hardware description languages. The processor is built with static CMOS circuits, which have been examined in Section 1.4 already; transistor level design and netlist formats are discussed. We continue exploring the physical design of the processor including floorplanning and area estimation. Design verification is very important and happens at each level of the hierarchy for each element of the design. Finally, the layout is converted into masks so the chip can be manufactured, packaged, and tested.

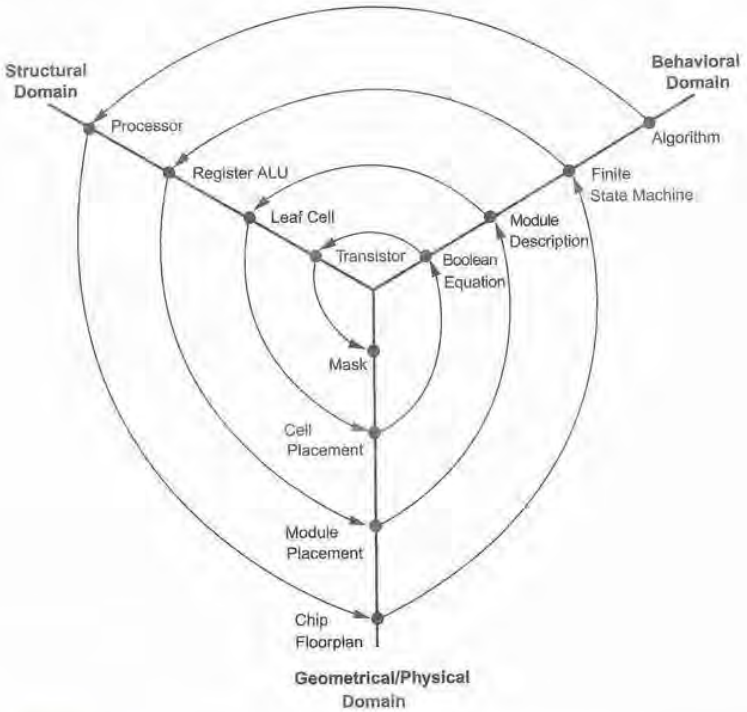


FIG 1.48 Y Diagram. Reproduced from [Kang03] with permission of The McGraw-Hill Companies.

1.7 Example: A Simple MIPS Microprocessor

We consider an 8-bit subset of the Patterson & Hennessy MIPS microprocessor architecture [Patterson04] because it is widely studied and is relatively simple, while still being large enough to illustrate hierarchical design. This section describes the architecture and the multicycle microarchitecture we will be implementing. If you are not familiar with computer architecture, you can regard the MIPS processor as a black box and skip to Section 1.8.

A set of laboratory exercises are available online in which you can learn VLSI design by building the microprocessor yourself using a free open-source CAD tool called *Electric*.

1.7.1 MIPS Architecture

The MIPS32 architecture is a simple 32-bit RISC architecture with relatively few idiosyncrasies. Our subset of the architecture uses 32-bit instruction encodings but only eight 8-bit general-purpose registers named `$0`–`$7`. We also use an 8-bit program counter (PC). Register `$0` is hardwired to contain the number 0. The instructions are `ADD`, `SUB`, `AND`, `OR`, `SLT`, `ADDI`, `BEQ`, `J`, `LB`, and `SB`.

The function and encoding of each instruction is given in Table 1.7. Each instruction is encoded using one of three templates: R, I, and J. R-type instructions (*register*-based) are used for arithmetic and specify two source registers and a destination register. I-type

Table 1.7 MIPS instruction set (subset supported)

Instruction	Function		Encoding	op	funct
`add $1, $2, $3`	addition:	`$1 <- $2 + $3`	R	000000	100000
`sub $1, $2, $3`	subtraction:	`$1 <- $2 - $3`	R	000000	100010
`and $1, $2, $3`	bitwise and:	`$1 <- $2 and $3`	R	000000	100100
`or $1, $2, $3`	bitwise or:	`$1 <- $2 or $3`	R	000000	100101
`slt $1, $2, $3`	set less than:	`$1 <- 1 if $2 < $3` `$1 <- 0 otherwise`	R	000000	101010
`addi $1, $2, imm`	add immediate:	`$1 <- $2 + imm`	I	001000	n/a
`beq $1, $2, imm`	branch if equal:	`PC <- PC + imm`[a]	I	000100	n/a
`j destination`	jump:	`PC <- destination`[a]	J	000010	n/a
`lb $1, imm($2)`	load byte:	`$1 <- mem[$2 + imm]`	I	100000	n/a
`sb $1, imm($2)`	store byte:	`mem[$2 + imm] <- $1`	I	101000	n/a

a. Technically, MIPS addresses specify bytes. Instructions require a four-byte word and must begin at addresses that are a multiple of four. To most effectively use instruction bits in the full 32-bit MIPS architecture, branch and jump constants are specified in words and must be multiplied by four (shifted left two bits) to be converted to byte addresses.

instructions are used when a 16-bit constant (also known as an *immediate*) and two registers must be specified. J-type instructions (*jumps*) dedicate most of the instruction word to a 26-bit jump destination. The format of each encoding is defined in Figure 1.49. The six most significant bits of all formats are the operation code (`op`). R-type instructions all share `op` = 000000 and use six more `funct` bits to differentiate the functions.

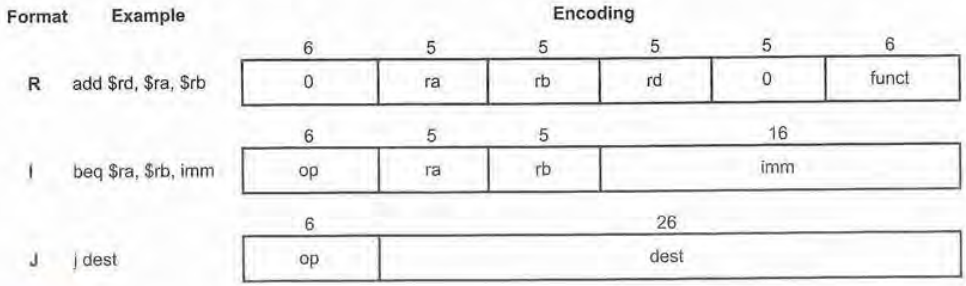


FIG 1.49 Instruction encoding formats

We can write programs for the MIPS processor in *assembly language*, where each line of the program contains one instruction such as `ADD` or `BEQ`. However, the MIPS hardware ultimately must read the program as a series of 32-bit numbers called *machine language*. An *assembler* automates the tedious process of translating from assembly language to machine language using the encodings defined in Table 1.7 and Figure 1.49. Writing nontrivial programs in assembly language is also tedious, so programmers usually work in a *high-level language* such as C or FORTRAN. A *compiler* translates a program from high-level language *source code* into the appropriate machine language *object code*.

Example

Figure 1.50 shows a simple C program that computes the nth Fibonacci number f_n defined recursively for $n > 0$ as $f_n = f_{n-1} + f_{n-2}, f_{-1} = -1, f_0 = 1$. Translate the program into MIPS assembly language and machine language.

Solution: Figure 1.51 gives a commented assembly language program. Figure 1.52 translates the assembly language to machine language.

continues

```
int fib(void)
{
   int n = 8;            /* compute nth Fibonacci number */
   int f1 = 1, f2 = -1;  /* last two Fibonacci numbers */

   while (n != 0) {      /* count down to n = 0 */
     f1 = f1 + f2;
     f2 = f1 - f2;
     n = n - 1;
   }
   return f1;
}
```

FIG 1.50 C code for Fibonacci program

```
# fib.asm
# Register usage: $3: n $4: f1 $5: f2
# return value written to address 255
fib:  addi $3, $0, 8       # initialize n=8
      addi $4, $0, 1       # initialize f1 = 1
      addi $5, $0, -1      # initialize f2 = -1
loop: beq $3, $0, end      # Done with loop if n = 0
      add $4, $4, $5       # f1 = f1 + f2
      sub $5, $4, $5       # f2 = f1 - f2
      addi $3, $3, -1      # n = n - 1
      j loop               # repeat until done
end:  sb $4, 255($0)       # store result in address 255
```

FIG 1.51 Assembly language code for Fibonacci program

Instruction	Binary Encoding	Hexadecimal Encoding
addi $3, $0, 8	001000 00000 00011 0000000000001000	20030008
addi $4, $0, 1	001000 00000 00100 0000000000000001	20040001
addi $5, $0, -1	001000 00000 00101 1111111111111111	2005ffff
beq $3, $0, end	000100 00011 00000 0000000000000101	10600005
add $4, $4, $5	000000 00100 00101 00100 00000 100000	00852020
sub $5, $4, $5	000000 00100 00101 00101 00000 100010	00852822
addi $3, $3, -1	001000 00011 00011 1111111111111111	2063ffff
j loop	000010 00000000000000000000000011	08000003
sb $4, 255($0)	110000 00000 00100 0000000011111111	a00400ff

FIG 1.52 Machine language code for Fibonacci program

1.7.2 Multicycle MIPS Microarchitecture

We will implement the multicycle MIPS microarchitecture given in Chapter 5 of [Patterson04] modified to process 8-bit data. The microarchitecture is illustrated in Figure 1.53. The rectangles represent registers or memory. The rounded rectangles represent multiplexers. The ovals represent control logic. Light lines indicate individual signals while heavy lines indicate busses. The control logic and signals are highlighted in blue while the datapath is shown in black. Control signals generally drive multiplexer select signals and register enables to tell the datapath how to execute an instruction.

Instruction execution generally flows from left to right. The program counter (PC) specifies the address of the instruction. The instruction is loaded one byte at a time over four cycles from an off-chip memory into the 32-bit instruction register (IR). The op field (bits 31:26 of the instruction) is sent to the controller, which sequences the datapath through the correct operations to execute the instruction. For example, in an ADD instruction, the two source registers are read from the register file into temporary registers a and b. On the next cycle, the alucontrol unit commands the Arithmetic/Logic Unit (ALU) to add the inputs. The result is captured in the `aluout` register. On the third cycle, the result is written back to the appropriate destination register in the register file.

The controller is a finite state machine that generates multiplexer select signals and register enables to sequence the datapath. A state transition diagram for the FSM is shown in Figure 1.54. As discussed, the first four states fetch the instruction from memory. The FSM then is dispatched based on op to execute the particular instruction. The FSM states for ADDI are missing and left as an exercise for the reader.

Observe that the controller produces a 2-bit `aluop` output. The alucontrol unit uses combinational logic to compute a 3-bit `alucontrol` signal from the `aluop` and `funct` fields, as specified in Table 1.8. `alucontrol` drives multiplexers in the ALU to select the appropriate computation.

Table 1.8 ALUControl determination

aluop	funct	alucontrol	Meaning
00	x	010	ADD
01	x	110	SUB
10	100000	010	ADD
10	100010	110	SUB
10	100100	000	AND
10	100101	001	OR
10	101010	111	SLT
11	x	x	undefined

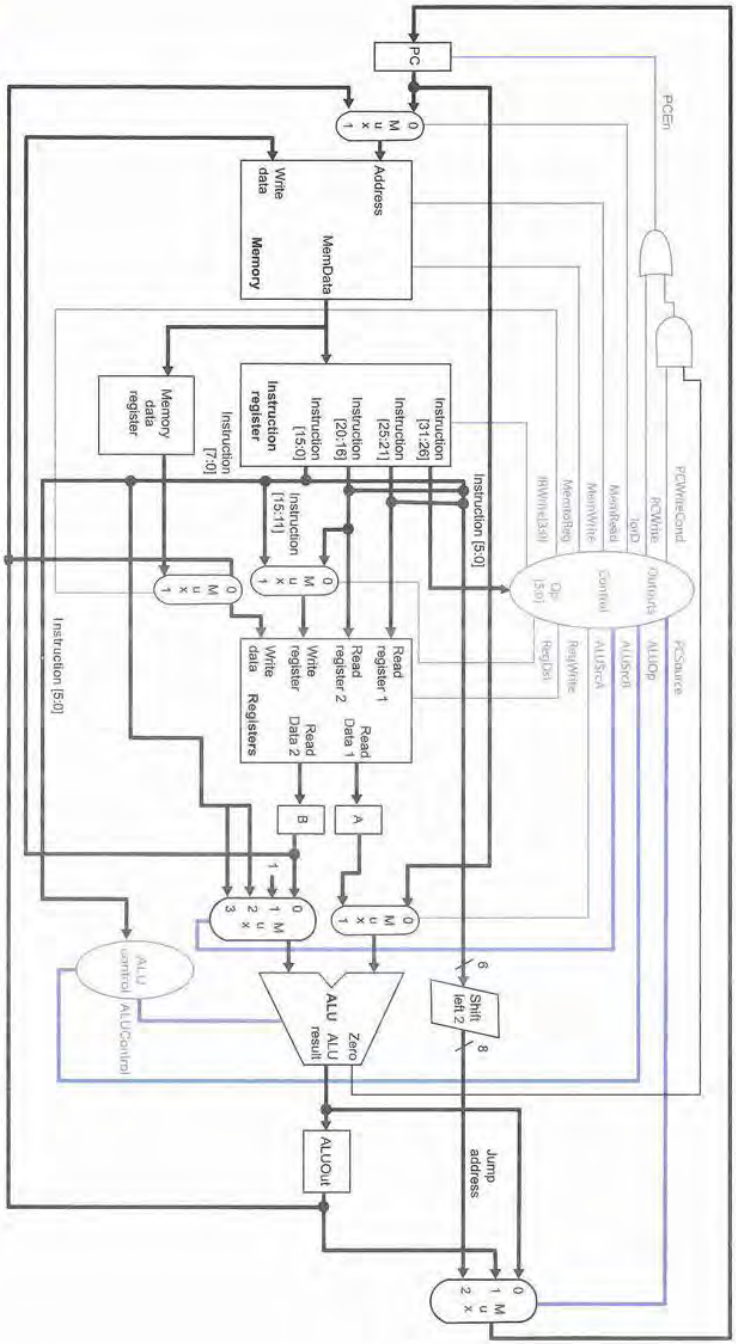


FIG 1.53 Multicycle MIPS microarchitecture. Reprinted from [Patterson04] with permission from Elsevier.

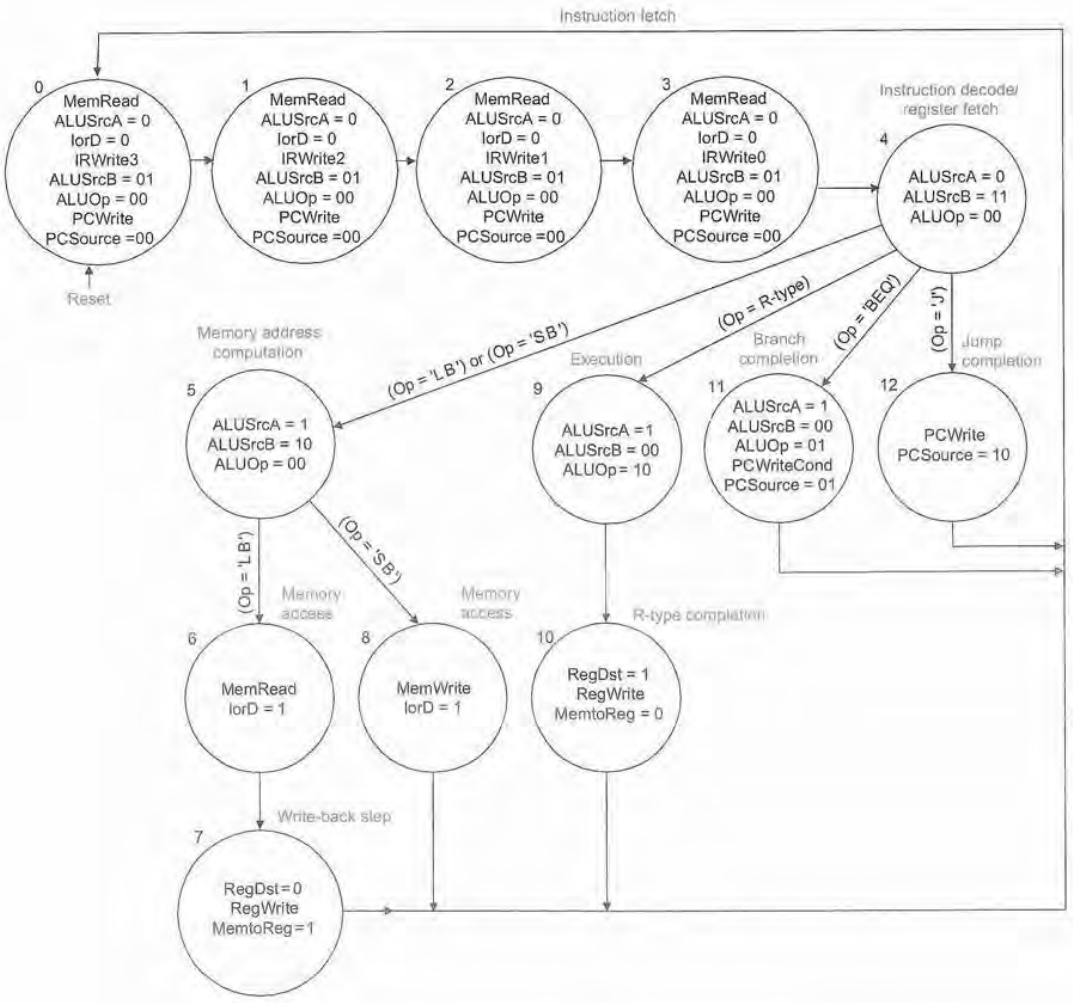


FIG 1.54 Multicycle MIPS control FSM. Reprinted from [Patterson04] with permission from Elsevier.

Example

Referring to Figure 1.53 and Figure 1.54, explain how the MIPS processor fetches and executes the SUB instruction.

Solution: The first step is to fetch the 32-bit instruction. This takes four cycles because the instruction must come over an 8-bit memory interface. On each cycle, we want to fetch a byte from the address in memory specified by the program counter, then increment the program counter by one to point to the next byte.

The fetch is performed by states 0–3 of the FSM in Figure 1.54. Let us start with state 0. The program counter (PC) contains the address of the first byte of the instruction. The controller must select `iord` = 0 so that the multiplexer sends this address to the memory. `memread` must also be asserted so the memory reads the byte onto the `memdata` bus. Finally, `irwrite3` should be asserted to enable writing `memdata` into the most significant byte of the instruction register (IR).

Meanwhile, we need to increment the program counter. We can do this with the ALU by specifying PC as one input, '1' as the other input, and ADD as the operation. To select PC as the first input, `alusrca` = 0. To select '1' as the other input, `alusrcb` = 01. To perform an addition, `aluop` = 00, according to Table 1.8. To write this result back into the program counter at the end of the cycle, `pcsource` = 00 and `pcen` = 1 (done by setting `pcwritecond` = 1).

All of these control signals are indicated in state 0 of Figure 1.54. The other register enables are assumed to be 0 if not explicitly asserted and the other multiplexer selects are don't cares. The next three states are identical except that they write bytes 2, 1, and 0 of the IR, respectively.

The next step is to read the source registers, done in state 4. The two source registers are specified in bits 25:21 and 20:16 of the IR. The register file reads these registers and puts the values into the A and B registers. No control signals are necessary for SUB (although state 4 performs a branch address computation in case the instruction is BEQ).

The next step is to perform the subtraction. Based on the `op` field (IR bits 31:26), the FSM jumps to state 9 because SUB is an R-type instruction. The two source registers are selected as input to the ALU by setting `alusrca` = 1 and `alusrcb` = 00. Choosing `aluop` = 10 directs the ALU Control decoder to select the `alucontrol` signal as 110, subtraction. Other R-type instructions are executed identically except that the decoder receives a different `funct` code (IR bits 5:0) and thus generates a different `alucontrol` signal. The result is placed in the ALUOut register.

Finally, the result must be written back to the register file in state 10. The data comes from the ALUOut register so `memtoreg` = 0. The destination register is specified in bits 15:11 of the instruction so `regdst` = 1. `regwrite` must be asserted to perform the write. Then the control FSM returns to state 0 to fetch the next instruction.

1.8 Logic Design

We begin the logic design by defining the top-level chip interface and block diagram. We then hierarchically decompose the units until we reach leaf cells. We specify the logic with a Hardware Description Language (HDL), which provides a higher level of abstraction than schematics or layout.

1.8.1 Top-level Interface

The top-level inputs and outputs are listed in Table 1.9. This example uses a two-phase clocking system to avoid hold-time problems. `Reset` initializes the PC to 0 and the control FSM to the start state. The remainder of the signals are used for an asynchronous 8-bit memory interface (assuming the memory is located off chip). The processor sends an 8-bit address `adr` and either asserts `memread` or `memwrite`. On a read cycle, the memory returns a value on the `memdata` lines while on a write cycle, the memory accepts input from `writedata`. In many systems, `memdata` and `writedata` can be combined onto a single bidirectional bus, but for this example we preserve the interface of Figure 1.53. Figure 1.55 shows a very simple computer system built from the MIPS processor, external memory, reset switch, and clock generator.

Table 1.9 Top-level inputs and outputs

Inputs	Outputs
`ph1`	`adr[7:0]`
`ph2`	`writedata[7:0]`
`reset`	`memread`
`memdata[7:0]`	`memwrite`

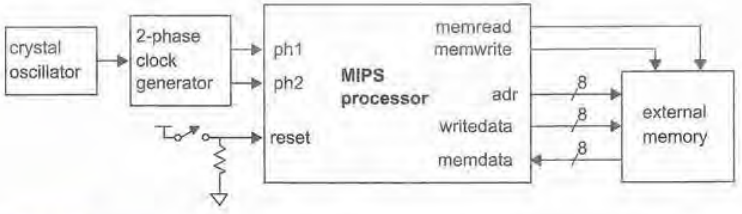


FIG 1.55 MIPS computer system

1.8.2 Block Diagram

The chip is partitioned into three top-level units: the controller, alucontrol, and datapath, as shown in the block diagram in Figure 1.56. The controller comprises the control FSM and the two gates used to compute `pcen`. The alucontrol consists of combinational logic to drive the ALU. The datapath contains the remainder of the chip, organized as eight identical *bitslices*. This partitioning is influenced by the intended physical design. The datapath contains most of the transistors and is very regular in structure. We can achieve high density with moderate design effort by handcrafting a single bitslice of the datapath, then replicating that bitslice eight times. The controller has much less structure. It is tedious to translate an FSM into gates by hand, and in a new design, the controller is the most likely portion to have bugs and last-minute changes. Therefore, we will specify the controller more abstractly with a hardware description language and automatically generate it using synthesis and place & route tools.

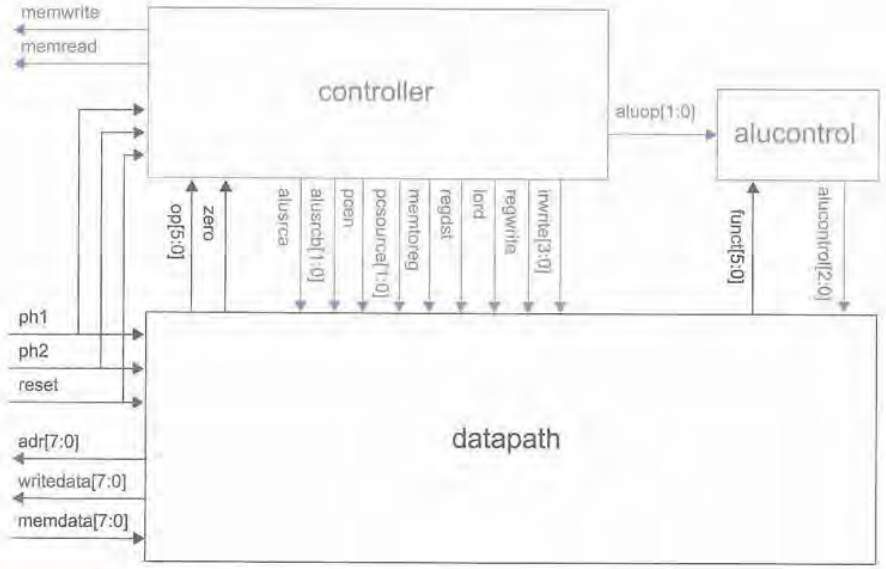


FIG 1.56 Top-level MIPS block diagram

1.8.3 Hierarchy

The best way to design complex systems is to decompose them into simpler pieces. Figure 1.57 shows the design hierarchy for the MIPS processor. The controller and alucontrol are built from a library of standard cells such as NANDs, NORs, and latches. The datapath is

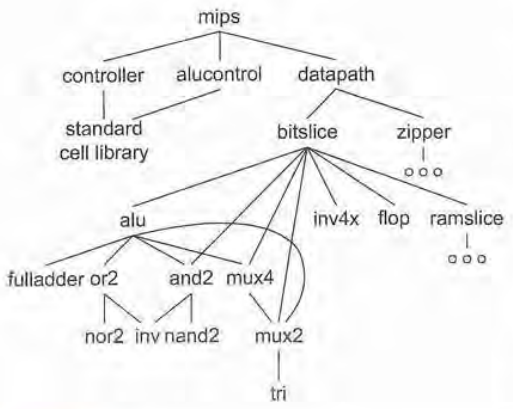


FIG 1.57 MIPS design hierarchy

composed of eight bitslices and a *zipper* that ties the datapath together by driving control signals and register enables to the various bits. The bitslice in turn is composed of the ALU, register file ramslice, flip-flops, and gates. Some of these gates are reused in multiple places. The pieces composing the zipper and ramslice are left out for brevity.

The design hierarchy does not necessarily have to be identical in the logic, circuit, and physical designs. For example, in the logic view, a memory may be best treated as a black box, while in the circuit implementation, it may have a decoder, cell array, column multiplexers, and so forth. Different hierarchies complicate verification, however, because they must be *flattened* until the point that they agree. As a matter of practice, it is best to make logic, circuit, and physical design hierarchies agree as far as possible.

1.8.4 Hardware Description Languages

Designers need rapid feedback on whether a logic design is reasonable. Translating a block diagram and finite state machine state transition diagrams into circuit schematics is time-consuming and prone to error; before going through this entire process it is wise to know if the top-level design has major bugs that will require complete redesign. HDLs provide a way to specify the design at a higher level of abstraction to raise designer productivity. They were originally intended for documentation and simulation, but are now used to synthesize gates directly from the HDL.

The two most popular HDLs are *Verilog* and *VHDL*. Verilog was developed by Advanced Integrated Design Systems (later renamed Gateway Design Automation) in 1984 and became a *de facto* industry open standard by 1991. VHDL, which stands for VHSIC Hardware Description Language, where VHSIC in turn was a Department of Defense project on Very High Speed Integrated Circuits, was developed by committee under government sponsorship. As one might expect from their pedigrees, Verilog is less verbose and closer in syntax to C, while VHDL supports some abstractions useful for large team projects. Many Silicon Valley companies use Verilog while defense and telecommunications companies often use VHDL. Neither language offers a compelling advantage over the other so the industry is saddled with supporting both. Appendices A and B offer brief tutorials on Verilog and VHDL. Examples in this book are given in Verilog for the sake of brevity.

When coding in an HDL, it is important to remember that you are specifying hardware that executes in parallel rather than software that executes sequentially. There are two general coding styles. *Structural* HDL specifies how a cell is composed of other cells or primitive gates and transistors. *Behavioral* HDL specifies what a cell does.

A *logic simulator* simulates both behavioral and structural HDL. A *logic synthesis* tool maps behavioral HDL code onto a *library* of gates called *standard cells* to minimize area while meeting some timing constraints. Only a subset of HDL constructs are synthesiz-

able; this subset is emphasized in the appendices. For example, file I/O commands used in testbenches are obviously not synthesizable. Logic synthesis generally produces circuits that are neither as dense nor as fast as those handcrafted by a skilled designer. Nevertheless, integrated circuit processes are now so advanced that synthesized circuits are good enough for the great majority of application-specific integrated circuits (ASICs) built today. Layout may be automatically generated using *place & route* tools.

Verilog and VHDL models for the MIPS processor are listed in Appendix A.10 and B.7. In Verilog, each cell is called a *module*. The inputs and outputs are declared much as in a C program and bit widths are given for busses. Internal signals must also be declared in a way analogous to local variables. The processor is described hierarchically using structural Verilog at the upper levels and behavioral Verilog for the leaf cells. For example, the controller module shows how a finite state machine is specified in behavioral Verilog and the alucontrol module shows how complex combinational logic is specified. The datapath is specified structurally, containing bitslices, which in turn contain an ALU, which in turn contains a full adder.

The full adder could be expressed structurally as a sum and a carry subcircuit. In turn, the sum and carry subcircuits could be expressed behaviorally. The full adder block is shown in Figure 1.58 while the carry subcircuit is explored further in Section 1.9.

```
module fulladder(input  a, b, c,
                 output s, cout);

   sum s1(a, b, c, s);
   carry c1(a, b, c, cout);
endmodule

module carry (input a, b, c,
              output cout)

   assign cout = (a&b) | (a&c) | (b&c);
endmodule
```

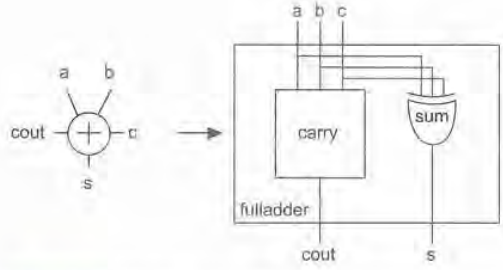


FIG 1.58 Full adder

1.9 Circuit Design

A particular logic function can be implemented in many ways. Should the function be built with ANDs, ORs, NANDs, or NORs? What should the fan-in and fan-out of each gate be? How wide should the transistors be on each gate? These and other choices influence the speed, power, and area of the system and are in the domain of circuit design.

As mentioned earlier, in many design methodologies, logic synthesis tools automatically make these choices, searching through the standard cells for the best implementation. For many applications, synthesis is good enough. When a system has critical requirements of high speed or low power or will be manufactured in large enough volume

to justify extra engineering to reduce die area, custom circuit design becomes important for critical portions of the chip.

Circuit designers often draw schematics at the transistor and/or gate level. For example, Figure 1.59 shows two alternative circuit designs for the carry circuit in a full adder. The gate-level design in Figure 1.59(a) requires 26 transistors and four stages of gate delays (recall that ANDs and ORs are built from NANDs and NORs followed by inverters). The transistor-level design in Figure 1.59(b) requires only 12 transistors and two stages of gate delays, illustrating the benefits of optimizing circuit designs to take advantage of CMOS technology.

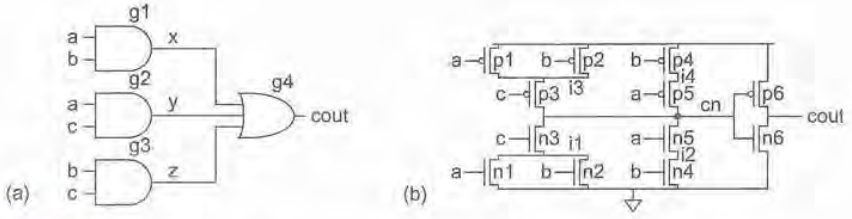


FIG 1.59 Carry subcircuit

These schematics are then *netlisted* for simulation and verification. One common netlist format is structural HDL. The gate-level design can be netlisted as shown below.

```
module carry(input  a, b, c,
             output cout)

  wire x, y, z;

  and g1(x, a, b);
  and g2(y, a, c);
  and g3(z, b, c);
  or g4(cout, x, y, z);
endmodule
```

This is a technology-independent structural description, because generic gates have been used and the actual gate implementations have not been specified. The transistor-level netlist is shown below.

```
module carry(input  a, b, c,
             output cout)
```

```
    wire i1, i2, i3, i4, cn;

    tranif1 n1(i1, 0, a);
    tranif1 n2(i1, 0, b);
    tranif1 n3(cn, i1, c);
    tranif1 n4(i2, 0, b);
    tranif1 n5(cn, i2, a);
    tranif0 p1(i3, 1, a);
    tranif0 p2(i3, 1, b);
    tranif0 p3(cn, i3, c);
    tranif0 p4(i4, 1, b);
    tranif0 p5(cn, i4, a);
    tranif1 n6(cout, 0, cn);
    tranif0 p6(cout, 1, cn);
endmodule
```

Transistors are expressed as

```
Transistor-type name(drain, source, gate);
```

`tranif1` corresponds to nMOS transistors that turn ON when the gate is '1' while `tranif0` corresponds to pMOS transistors that turn ON when the gate is '0.'

With the description generated so far, we still do not have the information required to determine the speed of the gate. We need to specify the size of the transistors and the stray capacitance. Because the Verilog language was designed as a switch-level and gate-level language, it is poorly suited to structural descriptions at this level of detail. Hence we turn to another common structural language used by the circuit simulator SPICE. The specification of the transistor-level carry subcircuit at the circuit level might be represented as shown below.

```
.SUBCKT CARRY A B C COUT VDD GND
MN1 I1 A GND GND NMOS W=1U L=0.18U AD=0.3P AS=0.5P
MN2 I1 B GND GND NMOS W=1U L=0.18U AD=0.3P AS=0.5P
MN3 CN C I1 GND NMOS W=1U L=0.18U AD=0.5P AS=0.5P
MN4 I2 B GND GND NMOS W=1U L=0.18U AD=0.15P AS=0.5P
MN5 CN A I2 GND NMOS W=1U L=0.18U AD=0.5P AS=0.15P
MP1 I3 A VDD VDD PMOS W=2U L=0.18U AD=0.6P AS=1 P
MP2 I3 B VDD VDD PMOS W=2U L=0.18U AD=0.6P AS=1P
MP3 CN C I3 VDD PMOS W=2U L=0.18U AD=1P AS=1P
MP4 I4 B VDD VDD PMOS W=2U L=0.18U AD=0.3P AS=1P
MP5 CN A I4 VDD PMOS W=2U L=0.18U AD=1P AS=0.3P
MN6 COUT CN GND GND NMOS W=2U L=0.18U AD=1P AS=1P
MP6 COUT CN VDD VDD PMOS W=4U L=0.18U AD=2P AS=2P
```

```
CI1 I1 GND 2FF
CI3 I3 GND 3FF
CA A GND 4FF
CB B GND 4FF
CC C GND 2FF
CCN CN GND 4FF
CCOUT COUT GND 2FF
.ENDS
```

Transistors are specified by lines beginning with an M as follows:

```
Mname   drain   gate   source   body   type   W=width   L=length
        AD=drain area   AS=source area
```

The body connection is new. Although MOS switches have been masquerading as three terminal devices (gate, source, and drain) until this point, they are in fact four terminal devices with the substrate or well acting as the body terminal. The type specifies whether the transistor is a p-device or n-device. The width, length, and area parameters specify physical dimensions of the actual transistors. Capacitors are specified by lines beginning with C as follows:

```
Cname   node1   node2   value
```

In this description the internal MOS model in SPICE calculates the parasitic capacitances inherent in the MOS transistor using the device dimensions specified. The extra capacitance statements in the above description designate additional routing capacitance not inherent to the device structure. This depends on the physical design of the gate. At the circuit level of structural specification, all connections are given that are necessary to fully characterize the carry gate in terms of speed, power, and connectivity.

1.10 Physical Design

1.10.1 Floorplanning

Physical design begins with a floorplan. The floorplan estimates the area of major units in the chip and defines their relative placements. The floorplan is essential to determine whether a proposed design will fit in the chip area budgeted and to estimate wiring lengths and wiring congestion, so an initial floorplan should be prepared as soon as the logic is loosely defined. As usual, this process involves feedback. The floorplan will often suggest changes to the logic (and microarchitecture), which in turn changes the floorplan. As a complex design begins to stabilize, the floorplan is often hierarchically subdivided to describe the functional blocks within the units.

Figure 1.60 shows the chip floorplan for the MIPS processor including the padframe. The top-level blocks are the controller, alucontrol, and datapath. A wiring channel is located between the controller and datapath to provide room to route the 30 control signals to the datapath. The datapath is further partitioned into the eight bitslices and the zipper. The padframe includes 40 I/O pads, which are wired to the pins on the chip package. There are 29 pads used for signals; the remainder are V_{DD} and GND.

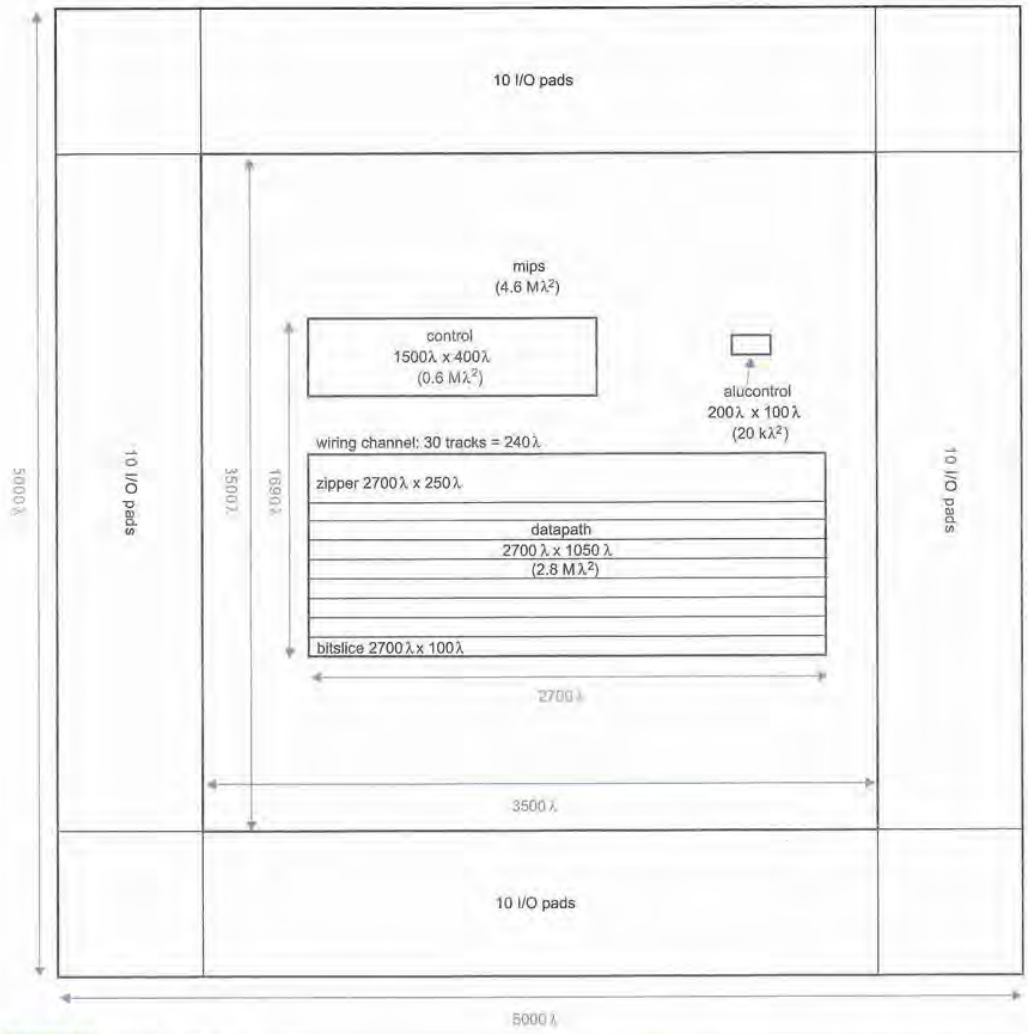


FIG 1.60 MIPS floorplan

The floorplan is drawn to scale and annotated with dimensions. The chip is designed in a 0.6 μm process on a 1.5 × 1.5 mm die so the die is 5000 λ on a side. The maximum possible core area inside the padframe is 3500 λ × 3500 λ = 12.25 Mλ². Due to the wiring channel and wasted space in the upper right corner, the actual core area of 4.6 Mλ² is larger than the sum of the block areas. This design is said to be *pad-limited* because the I/O pads set the chip area. Most commercial chips are *core-limited* because the chip area is set by the logic excluding the pads. In general, blocks in a floorplan should be rectangular because it is difficult for a designer to stuff logic into an odd-shaped region.

Figure 1.61 shows the actual chip layout. Notice the 40 I/O pads around the periphery. Just inside the pad frame are metal2 V_{DD} and GND rings, marked with + and −. For simplicity, the routing between the padframe and core is not shown.

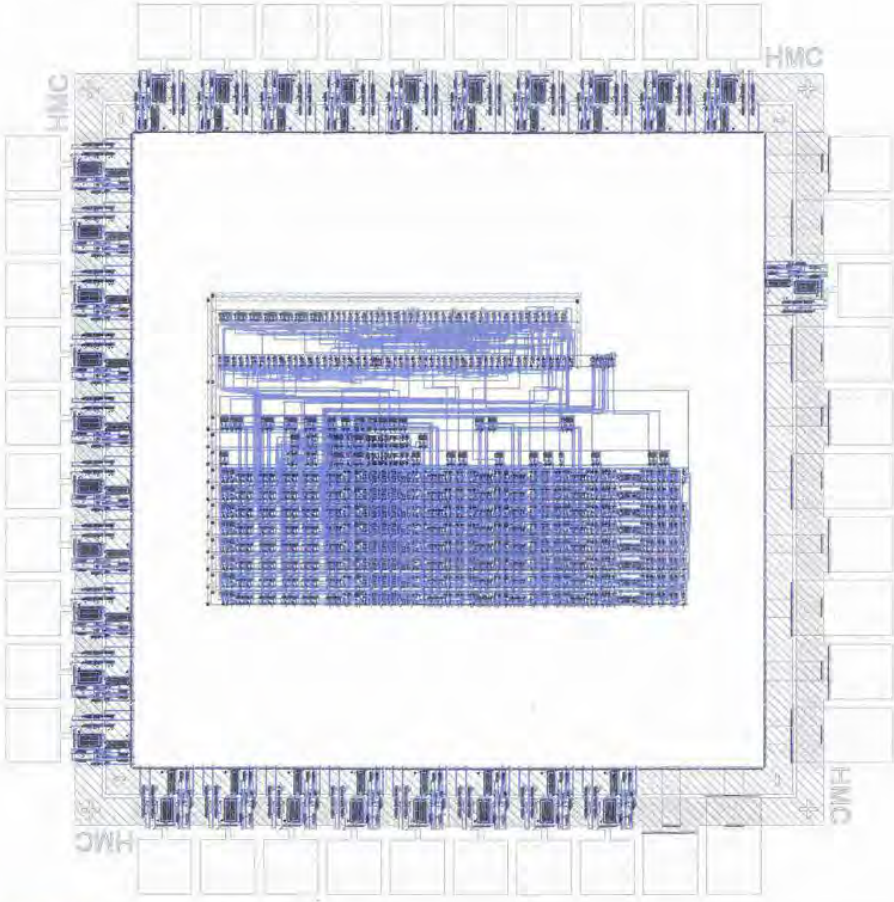


FIG 1.61 MIPS layout

On-chip structures can be categorized as *random logic*, *datapaths*, *arrays*, *analog*, and *input/output* (I/O). Random logic, like the controller, has little structure. Datapaths operate on multi-bit data words and perform roughly the same function on each bit so they consist of multiple identical or nearly identical bitslices repeated in one dimension. Arrays, like RAMs, ROMs, and PLAs, consist of identical cells repeated in two dimensions. Productivity is highest if layout can be reused or automatically generated. Datapaths and arrays are good VLSI building blocks because a single carefully crafted cell is reused in one or two dimensions. Automatic layout generators exist for memory arrays and random logic but are not as mature for datapaths. Therefore, many design methodologies ignore the potential structure of datapaths and instead lay them out with random logic tools except when performance or area are vital. Analog circuits still require careful design and simulation but tend to involve only small amounts of layout. I/O cells are also highly tuned to each fabrication process and are often supplied by the process vendor.

The MIPS layout demonstrates two of the most common layout styles: *standard cells* and *snap-together cells*. Standard cells are used for random logic. Snap-together cells are used in datapaths and arrays. Each is discussed in the following sections.

1.10.2 Standard Cells

A simple standard cell library for logic synthesis is shown in Figure 1.62 and the inside front cover. The standard cells all share the same metal1 V_{DD} and GND locations and 60 λ height so that power and ground can be connected by abutment. Inputs and outputs are provided in metal2.

In a process with two or three metal layers, standard cells are tiled into rows separated by *routing channels*. The number of wires that must be routed sets the height of the routing channels. Layout is often generated with automatic place & route tools. Figure 1.63 shows the controller layout generated by such a tool. In this example, metal1 is used horizontally and metal2 is used vertically. When more layers of metal are available, routing takes place over the cells and routing channels become unnecessary. Note that in this and subsequent layouts, the n-well around the pMOS transistors will usually not be shown.

Automatic synthesis and place & route tools have become good enough to map entire designs onto standard cells. Figure 1.64 shows the entire 8-bit MIPS processor synthesized from the VHDL model given in Appendix B onto a cell library in a 130 nm process with seven metal layers. Compared to Figure 1.61, the synthesized design shows no discernable structure. Synthesized designs tend to be somewhat larger and slower than a good custom design, but they also take an order of magnitude less design time.

1.10.3 Snap-together Cells

The area of the controller in Figure 1.63 is dominated by the routing channels. When the logic is more regular, layout density can be improved by including the wires in cells that "snap together." Snap-together cells require more design and layout effort but lead to smaller area and shorter (i.e., faster) wires. The key issue in designing snap-together cells is *pitch-matching*. Cells that connect together must have the same size in the connecting edge. Figure 1.65 shows several pitch-matched cells. Reducing the size of cell *D* does not

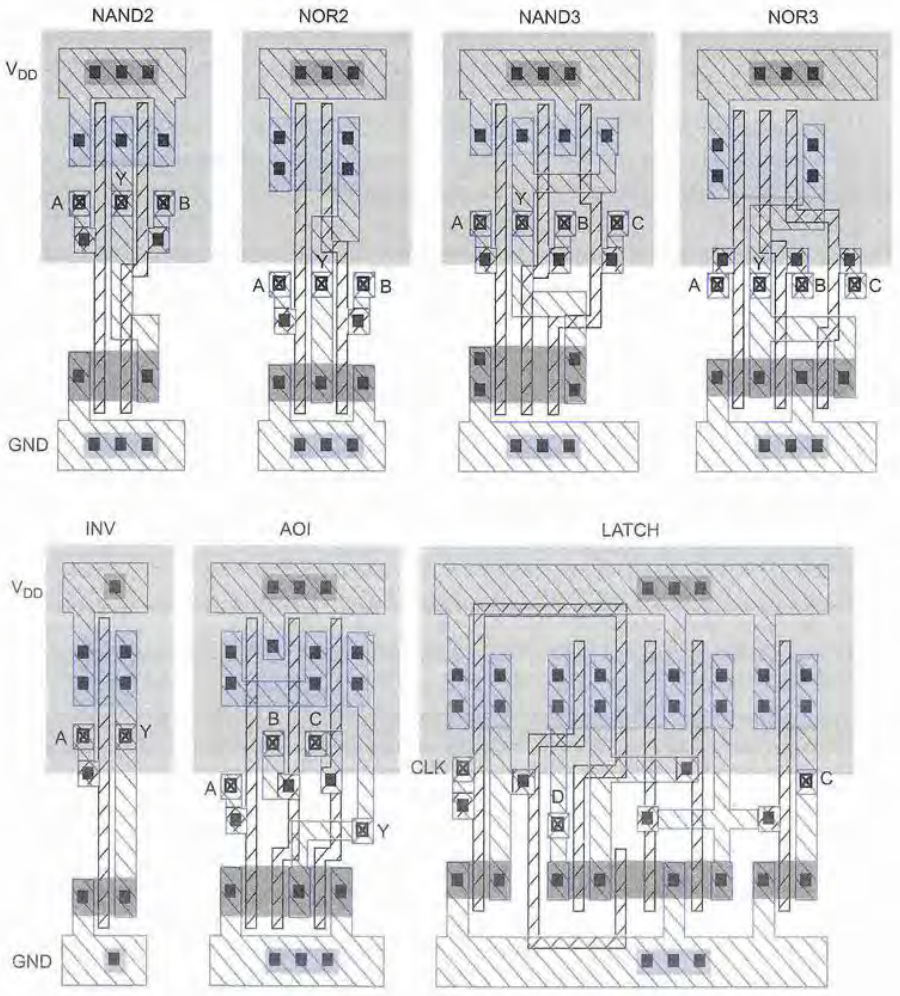


FIG 1.62 Simple standard cell library. Color version on inside front cover.

help the layout area. On the other hand, increasing the size of cell *D* also affects the area of *B* and/or *C*.

Figure 1.66 shows the MIPS datapath in more detail. The eight horizontal bitslices are clearly visible. The zipper at the top of the layout includes a decoder that is pitch-matched to the register file in the datapath.

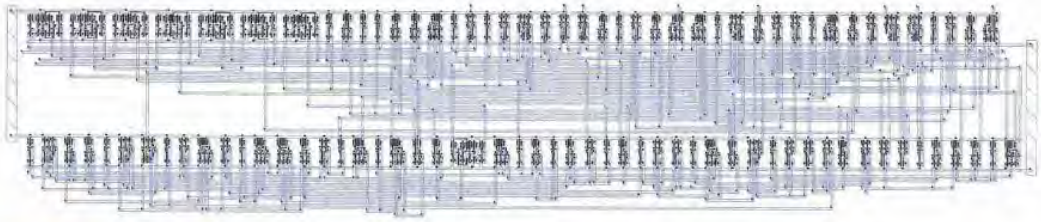


FIG 1.63 MIPS controller layout

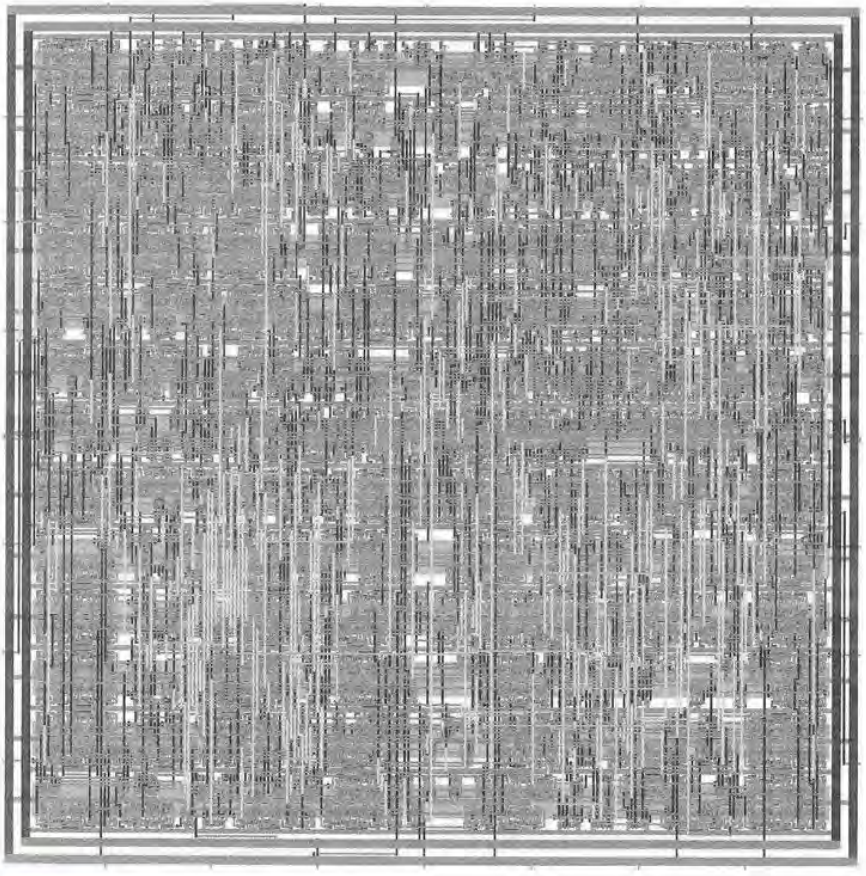


FIG 1.64 Synthesized MIPS processor

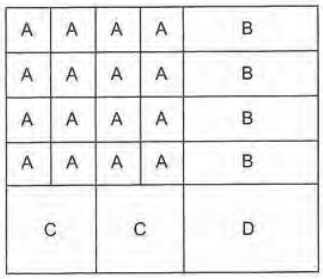


FIG 1.65 Pitch-matching of snap-together cells

The number of wires that must run over the datapath determines the height of the datapath cells. 80–100 λ are typical heights for relatively simple datapaths. The width of the cell depends on the cell contents. In this layout, metal1 is used for local wiring within a cell. V_{DD} and GND are routed horizontally in metal2. Horizontal metal2 wires also run over the tops of cells to carry data between cells in a bitslice. Vertical metal3 control wires are used for control, including clocks, multiplexer selects, and register enables.

Figure 1.67 shows a blowup of the ALU at the right end of each bitslice. Observe how horizontal metal2 data wires and vertical metal3 multiplexer select lines are routed over the top of the cells so no routing channels are required. The adder is shown as a black box and its design is left as an exercise for the reader.

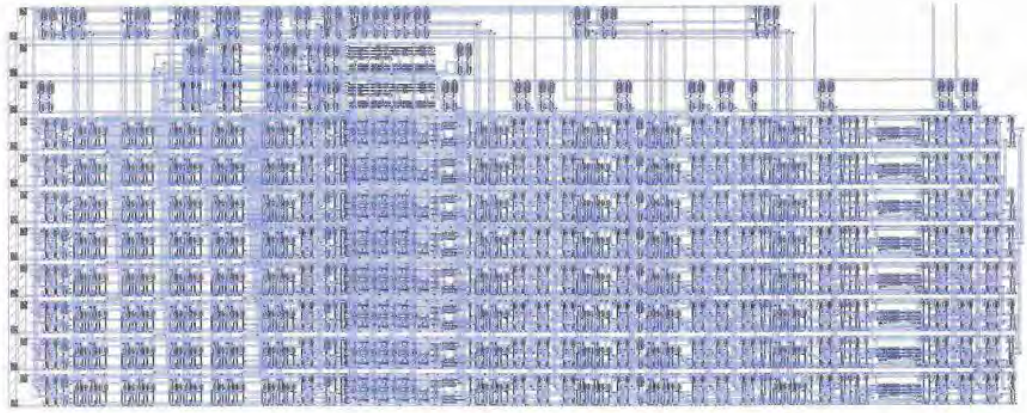


FIG 1.66 MIPS datapath layout

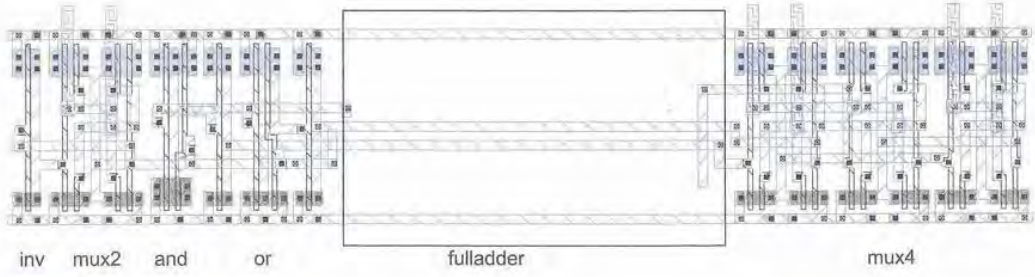


FIG 1.67 MIPS ALU layout

1.10.4 Slice Plans

Figure 1.68 shows a *slice plan* for a bitslice of the datapath. The diagram illustrates the ordering of elements in the bitslice and the allocation of wiring tracks. Wires at the left side of a cell are inputs and wires at the right side are outputs. Dots indicate that a wire passes over a cell and is also used in that cell. Each cell is annotated with its type and width. For example, the program counter (pc) is an output of the PC flop and is also used as an input to the srcA and address multiplexers. The slice plan makes it easy to calculate wire lengths and evaluate wiring congestion before laying out the datapath. In this case, it is evident that the greatest congestion takes place over the register file, where eight wiring tracks are required. The slice plan is also useful for estimating area of datapaths.

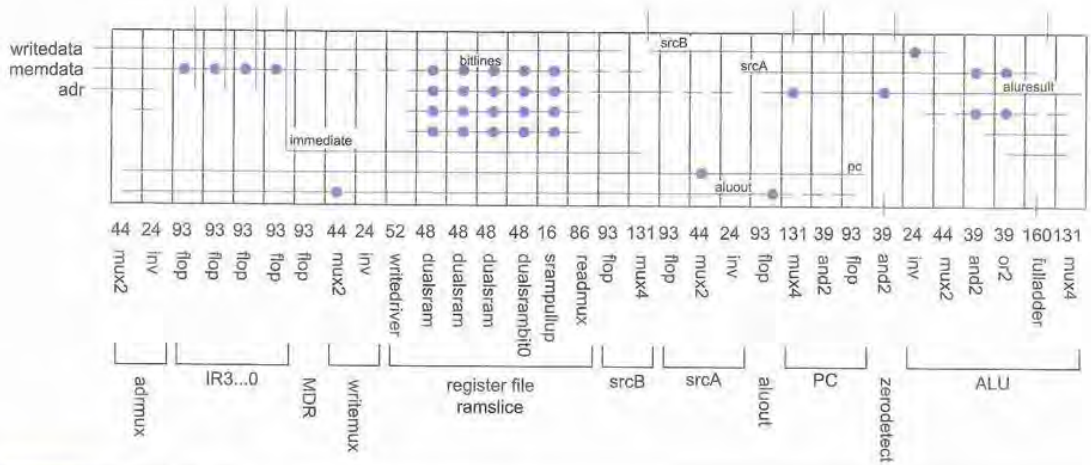


FIG 1.68 Datapath slice plan

1.10.5 Area Estimation

A good floorplan depends on reasonable area estimates, which may be difficult to make before logic is finalized. An experienced designer may be able to estimate block area by comparison to the area of a comparable block drawn in the past. In the absence of data for such comparison, Table 1.10 lists some typical numbers. Be certain to account for large wiring channels at a pitch of 8 λ / track. Larger transistors clearly occupy greater area, so this may be factored into the area estimates as a function of W and L (width and length) [Horowitz02]. Your mileage may vary, but it is clear that datapaths and arrays achieve much higher densities than standard cells. In the MIPS example, the controller random logic has 440 transistors in 0.6 $M\lambda^2$ for a density of about 1400 λ^2/transistor. The datapath has 3461 transistors in a 2.7 $M\lambda^2$ area for a density of 780 λ^2/transistor.

Table 1.10 Typical layout densities

Element	Area
random logic (2-level metal process)	1000 – 1500 λ^2 / transistor
datapath	250 – 750 λ^2 / transistor or 6 WL + 360 λ^2 / transistor
SRAM	1000 λ^2 / bit
DRAM (in a DRAM process)	100 λ^2 / bit
ROM	100 λ^2 / bit

Given enough time, it is nearly always possible to shave a few lambda here or there from a design. However, such efforts are seldom a good use of time unless an element is repeated so many times that it accounts for a major fraction of the chip area or if floorplan errors have led to too little space for a block and the block must be shrunk before the chip can be completed. It is wise to make conservative area estimates in floorplans, especially if there is risk that more functionality may be added to a block.

Some cell library vendors specify typical routed standard cell layout densities in kgates/mm^2.[6] Commonly a gate is defined as a 3-input static CMOS NAND or NOR with six transistors. A 180 nm process ($\lambda \sim 0.1$ μm) with six metal layers may achieve a density of 20 kgates/mm^2 for random logic. This corresponds to about 830 λ^2 / transistor.

1.11 Design Verification

Integrated circuits are complicated enough that it is safe to assume that if anything can go wrong, it probably has. Design verification is essential to catching the errors before manufacturing and commonly accounts for one-third to one-half of the effort devoted to a chip.

As design representations become more detailed, verification time increases. It is not practical to simulate an entire chip in a circuit-level simulator such as SPICE for a large number of cycles to prove that the layout is correct. Instead, the design is usually tested for functionality at the architectural level with a model in a language such as C and at the logic level by simulating the HDL description. Then the circuits are checked to ensure they are a faithful representation of the logic and the layout is checked to ensure it is a faithful representation of the circuits, as shown in Figure 1.69. Circuits and layout must meet timing and power specifications as well.

A *testbench* is used to verify that the logic is correct. The testbench instantiates the logic under test. It reads a file of inputs and expected outputs called *test vectors*, applies them to the module under test, and logs mismatches. Appendix A.10 provides an example of a testbench for verifying the MIPS processor logic.

[6] 1 kgate = 1000 gates

A number of techniques are available for circuit verification. If the logic is synthesized onto a cell library, the postsynthesis gate-level netlist can be expressed in an HDL again and simulated using the same test vectors. Powerful *formal verification* tools are also becoming available that prove that a circuit performs the same Boolean function as the associated logic. These are very useful for handcrafted circuits, but can be slow and difficult to operate. Exotic circuits should be simulated thoroughly to ensure they perform the intended logic function and have adequate noise margins; circuit pitfalls are discussed throughout this book.

Layout vs. Schematic tools (LVS) check to make sure that transistors in a layout are connected in the same way as in the circuit schematic. *Design rule checkers* (DRC) verify that the layout satisfies design rules. *Electrical rule checkers* (ERC) scan for other potential problems such as latch-up, noise problems, or electromigration risks; such problems will also be discussed later in the book.

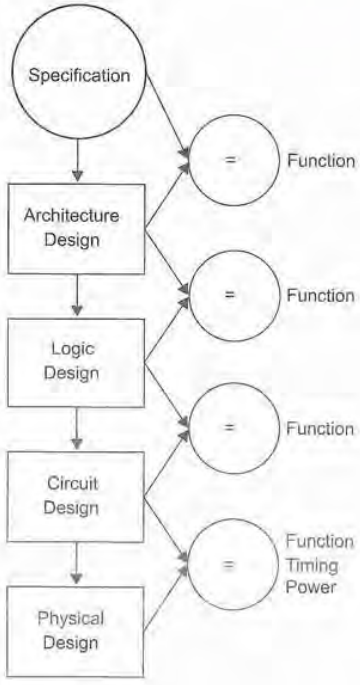


FIG 1.69 Design and verification sequence

1.12 Fabrication, Packaging, and Testing

Once a chip design is complete, it is taped out for manufacturing. *Tapeout* gets its name from the old practice of writing a specification of masks to magnetic tape; now the mask descriptions are usually sent to the manufacturer electronically. Two common formats for mask descriptions are the Caltech Interchange Format (CIF) [Mead80] (mainly used in academia) and the Calma GDS II Stream Format (GDS) [Calma84] (used in industry).

Masks are made by etching a pattern of chrome on glass with an electron beam. A set of masks for a modern process can be very expensive. For example, masks for a large chip in a 180 nm process may cost on the order of a quarter of a million dollars. In a 130 nm process, the mask set may be in the vicinity of a million dollars. The MOSIS service in the United States and its counterparts in Europe and Japan make a single set of masks covering multiple small designs from academia and industry to amortize the cost across many customers.

Integrated circuit fabrication plants (fabs) now cost billions of dollars and become obsolete in a few years. Some large companies still own their own fabs, but an increasing number of fabless semiconductor companies contract out manufacturing to vendors such as TSMC, UMC, and IBM.

Multiple chips are manufactured simultaneously on a single silicon wafer, typically 6″–12″ in diameter. A bare wafer costs about $1000–$5000. Fabrication requires many deposition, masking, etching, and implant steps. Most fabrication plants are optimized for wafer throughput rather than latency, leading to turnaround times of up to 10 weeks, although shorter turnarounds are available for a substantial premium. Figure 1.70 shows a silicon wafer after processing.

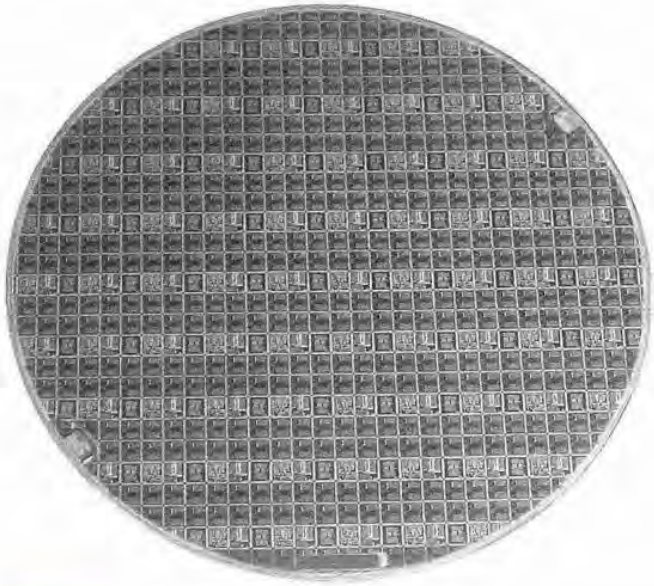


FIG 1.70 Processed 8-inch wafer

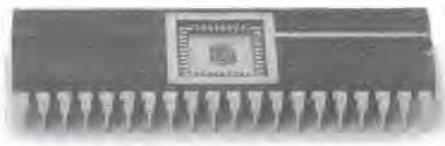


FIG 1.71 Chip in a 40-pin dual-inline package

Processed wafers are sliced into dice (chips) and packaged. Figure 1.71 shows a 1.5 × 1.5 mm chip in a 40-pin *dual-inline package* (DIP). This *wire-bonded* package uses thin gold wires to connect the pads on the die to the lead frame in the center cavity of the package. More advanced packages offer different tradeoffs between cost, pin count, pin bandwidth, power handling, and reliability, as will be discussed in Section 12.2. Flip-chip technology places small solder balls directly onto the die, eliminating the bond wire inductance and allowing contacts over the entire chip area rather than just at the periphery.

Even tiny defects in a wafer or dust particles can cause a chip to fail. Chips are tested before being sold. Testers capable of handling high-speed chips cost millions of dollars, so many chips use built-in self-test features to reduce the tester time required. Chapter 9 is devoted to design verification and testing.

Summary

"If the automobile had followed the same development cycle as the computer, a Rolls-Royce would today cost $100, get one million miles to the gallon, and explode once a year . . ."

—Robert X. Cringely

CMOS technology, driven by Moore's Law, has come to dominate the semiconductor industry. This chapter developed the principles of designing a simple CMOS integrated circuit. MOS transistors can be viewed as electrically controlled switches. Complementary CMOS gates are built from pull-down networks of nMOS transistors and pull-up networks of pMOS transistors. Transistors and wires are fabricated on silicon wafers using a series of deposition, lithography, and etch steps. These steps are defined by a set of masks drawn as a chip layout. Design rules specify minimum width and spacing between elements in the layout. The chip design process can be divided into architecture, logic, circuit, and physical design. The performance, area, and power of the chip are influenced by interrelated decisions made at each level. Design verification plays an important role in constructing such complex systems; the reliability requirements for hardware are much greater than those typically imposed on software. The remainder of this book will expand on the material introduced in this chapter.

Exercises

1.1 Extrapolating the data from Figure 1.4, predict the transistor count of a microprocessor in 2010.

1.2 Search the Web for transistor counts of Intel's more recent microprocessors. Make a graph of transistor count V_s. year of introduction from the Pentium Processor in 1993 to the present on a semilog scale. How many months pass between doubling of transistor counts?

1.3 Sketch a transistor-level schematic for a CMOS 4-input NOR gate.

1.4 Sketch a transistor-level schematic for a single-stage CMOS logic gate for each of the following functions:

a) $Y = \overline{ABC + D}$

b) $Y = \overline{(AB + C) \cdot D}$

c) $Y = \overline{AB + C \cdot (A + B)}$

1.5 Use a combination of CMOS gates (represented by their symbols) to generate the following functions from A, B, and C.

a) $Y = A$ (buffer)

b) $Y = A\overline{B} + \overline{A}B$ (XOR)

c) $Y = \overline{A\overline{B} + AB}$ (XNOR)

d) $Y = AB + BC + AC$ (majority)

1.6 Sketch a transistor-level schematic of a CMOS 3-input XOR gate. You may assume you have both true and complementary versions of the inputs available.

1.7 Sketch transistor-level schematics for the following logic functions. You may assume you have both true and complementary versions of the inputs available.

a) A 2:4 decoder defined by

$$Y0 = \overline{A0} \cdot \overline{A1}$$
$$Y1 = A0 \cdot \overline{A1}$$
$$Y2 = \overline{A0} \cdot A1$$
$$Y3 = A0 \cdot A1$$

b) A 3:2 priority encoder defined by

$$Y0 = \overline{A0} \cdot (A1 + \overline{A2})$$
$$Y1 = \overline{A0} \cdot \overline{A1}$$

1.8 Sketch a stick diagram for a CMOS 4-input NOR gate from Exercise 1.3.

1.9 Estimate the area of your 4-input NOR gate from Exercise 1.8.

1.10 Using a CAD tool of your choice, layout a 4-input NOR gate. How does its size compare to the prediction from Exercise 1.9?

1.11 Figure 1.72 shows a stick diagram of a 2-input NAND gate. Sketch a side view (cross-section) of the gate from X to X'.

1.12 Figure 1.73 gives a stick diagram for a level-sensitive latch. Estimate the area of the latch.

1.13 Draw a transistor-level schematic for the latch of Figure 1.73. How does the schematic differ from Figure 1.30(b)?

1.14 Consider the design of a CMOS compound OR-AND-INVERT (OAI21) gate computing $F = \overline{(A + B) \cdot C}$.

a) sketch a transistor-level schematic

b) sketch a stick diagram

c) estimate the area from the stick diagram

d) layout your gate with a CAD tool

e) compare the layout size to the estimated area

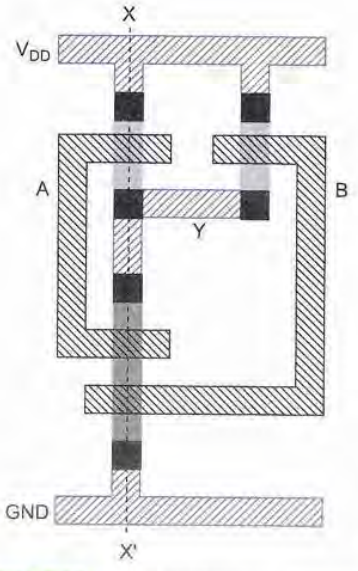


FIG 1.72 2-input NAND gate stick diagram

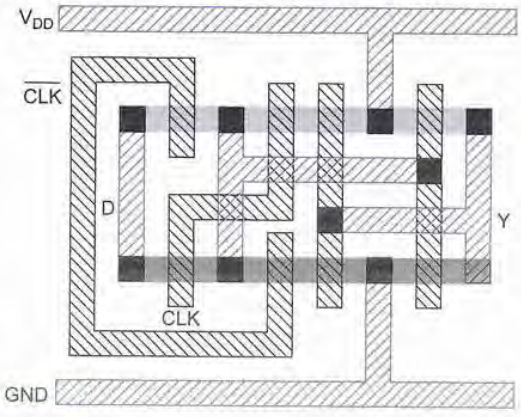


FIG 1.73 Level-sensitive latch stick diagram

1.15 Consider the design of a CMOS compound OR-OR-AND-INVERT (OAI22) gate computing $F = \overline{(A + B) \cdot (C + D)}$.

a) sketch a transistor-level schematic

b) sketch a stick diagram

c) estimate the area from the stick diagram

d) layout your gate with a CAD tool

e) compare the layout size to the estimated area

1.16 A 3-input majority gate returns a true output if at least two of the inputs are true. A minority gate is its complement. Design a 3-input CMOS minority gate using a single stage of logic.

a) sketch a transistor-level schematic

b) sketch a stick diagram

c) estimate the area from the stick diagram

1.17 Design a 3-input minority gate using CMOS NANDs, NORs, and inverters. How many transistors are required? How does this compare to a design from Exercise 1.16(a)?

1.18 A carry lookahead adder computes $G = G_3 + P_3(G_2 + P_2(G_1 + P_1G_0))$. Consider designing a compound gate to compute G.

a) sketch a transistor-level schematic

b) sketch a stick diagram

c) estimate the area from the stick diagram

1.19 The course Web page has a series of five labs in which you can learn VLSI design by completing the multicycle MIPS processor described in this chapter. The labs use the open-source Electric CAD tool. They cover:

a) leaf cells: schematic entry, layout, icons, simulation, DRC, ERC, LVS; hierarchical design

b) complex leaf cell design and verification: full adder

c) hierarchical design: ALU assembly, datapath routing, and simulation

d) control design: Verilog, synthesis, place & route

e) chip assembly, pad frame, full-chip verification, tapeout

MOS Transistor Theory

2

2.1 Introduction

In Chapter 1 the Metal-Oxide-Semiconductor (MOS) transistor was introduced in terms of its operation as an ideal switch. In this chapter we will examine the characteristics of MOS transistors in more detail to lay the foundation for predicting their performance. We will also look at second-order effects that are important to power consumption and circuit reliability. Using the transistor models, we will calculate the DC transfer characteristics of logic gates.

Figure 2.1 shows some of the symbols that are commonly used for MOS transistors. The symbols in Figure 2.1(a) will be used where it is only necessary to indicate the switch logic necessary to build a function. If the body (substrate or well) connection needs to be shown, the symbols in Figure 2.1(b) will be used. Figure 2.1(c) shows an example of other symbols that may be encountered in the literature.

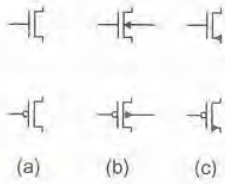


FIG 2.1 MOS transistor symbols

The MOS transistor is a *majority-carrier* device in which the current in a conducting channel between the source and drain is controlled by a voltage applied to the gate. In an nMOS transistor, the majority carriers are electrons; in a pMOS transistor, the majority carriers are holes. The behavior of MOS transistors can be understood by first examining an isolated MOS structure with a gate and body but no source or drain. Figure 2.2 shows a simple MOS structure. The top layer of the structure is a good conductor called the *gate*. Early transistors used metal gates, but modern transistors generally use polysilicon, i.e., silicon formed from many small crystals. The middle layer is a very thin insulating film of SiO_2 called the *gate oxide*. The bottom layer is the doped silicon body. The figure shows a p-type body in which the carriers are holes. The body is grounded and a voltage is applied to the gate. The gate oxide is a good insulator so almost zero current flows from the gate to the body.[1]

In Figure 2.2(a), a negative voltage is applied to the gate, so there is negative charge on the gate. The mobile positively charged holes are attracted to the region beneath the gate. This is called the *accumulation* mode. In Figure 2.2(b), a low positive voltage is applied to the gate, resulting in some positive charge on the gate. The holes in the body are repelled from the region directly beneath the gate, resulting in a *depletion* region forming below the gate. In Figure 2.2(c), a higher positive potential exceeding a critical threshold voltage V_t is

[1]Gate oxides are now only a handful of atomic layers thick and electrons sometimes tunnel through the oxide, creating a current from gate to substrate. This effect is explored in Section 2.4.5.

applied, attracting more positive charge to the gate. The holes are repelled further and a small number of free electrons in the body are attracted to the region beneath the gate. This conductive layer of electrons in the p-type body is called the *inversion* layer. The threshold voltage depends on the number of dopants in the body and the thickness t_{ox} of the oxide. It is usually positive, as shown in this example, but can be engineered to be negative.

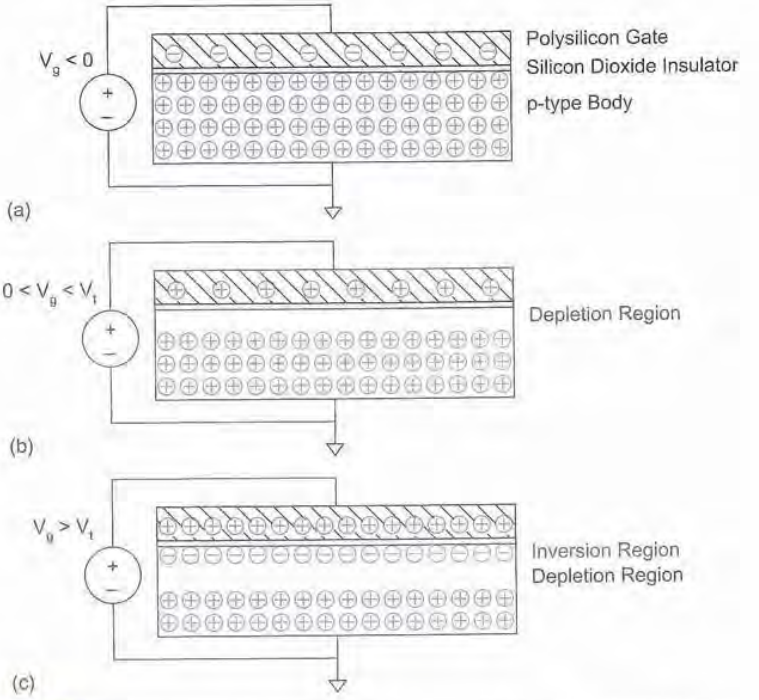


FIG 2.2 MOS structure demonstrating (a) accumulation, (b) depletion, and (c) inversion

Figure 2.3 shows an nMOS transistor with a grounded source and p-type body. The transistor consists of the MOS stack between two n-type regions called the *source* and *drain*. In Figure 2.3(a), the gate-to-source voltage V_{gs} is less than the threshold voltage. The source and drain have free electrons. The body has free holes but no free electrons. The junctions between the body and the source or drain are reverse-biased, so almost zero current flows. This mode of operation is called *cutoff*. In Figure 2.3(b), the gate voltage is greater than the threshold voltage. Now an inversion region of electrons (majority carriers) called the *channel* connects the source and drain, creating a conductive path. The number of carriers and the conductivity increases with the gate voltage. The potential difference

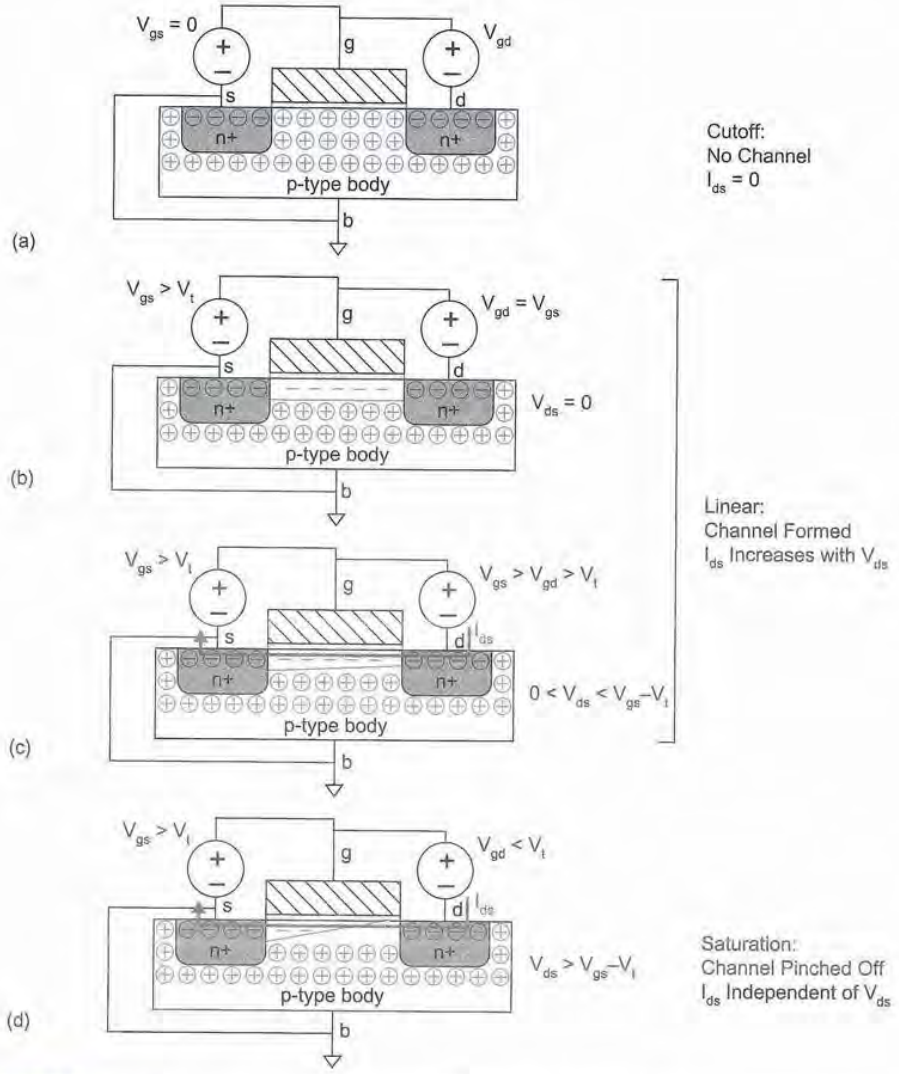


FIG 2.3 nMOS transistor demonstrating cutoff, linear, and saturation regions of operation

between drain and source is $V_{ds} = V_{gs} - V_{gd}$. If $V_{ds} = 0$ (i.e., $V_{gs} = V_{gd}$), there is no electric field tending to push current from drain to source. When a small positive potential V_{ds} is applied to the drain (Figure 2.3(c)), current I_{ds} flows through the channel from drain to

source[2]. This mode of operation is termed *linear*, *resistive*, *nonsaturated*, or *unsaturated*; the current increases with both the drain voltage and gate voltage. If V_{ds} becomes sufficiently large that $V_{gd} < V_t$, the channel is no longer inverted near the drain and becomes *pinched off* (Figure 2.3(d)). However, conduction is still brought about by the drift of electrons under the influence of the positive drain voltage. As electrons reach the end of the channel, they are injected into the depletion region near the drain and accelerated toward the drain. Above this drain voltage the current I_{ds} is controlled only by the gate voltage and ceases to be influenced by the drain. This mode is called *saturation*.

In summary, the nMOS transistor has three modes of operation. If $V_{gs} < V_t$, the transistor is cut off and no current flows. If $V_{gs} > V_t$ and V_{ds} is small, the transistor acts as a linear resistor in which the current flow is proportional to V_{ds}. If $V_{gs} > V_t$ and V_{ds} is large, the transistor acts as a current source in which the current flow becomes independent of V_{ds}.

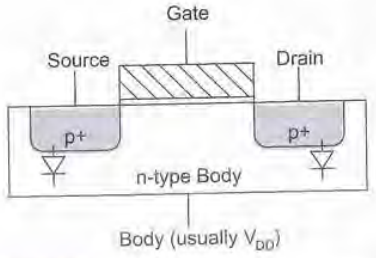


FIG 2.4 pMOS transistor

The pMOS transistor in Figure 2.4 operates in just the opposite fashion. The n-type body is tied to a high potential so the junctions with the p-type source and drain are normally reverse-biased. When the gate is also at a high potential, no current flows between drain and source. When the gate voltage is lowered by a threshold V_t, holes are attracted to form a p-type channel immediately beneath the gate, allowing current to flow between drain and source. The threshold voltages of the two types of transistors are not necessarily equal, so we use the terms V_{tn} and V_{tp} to distinguish the nMOS and pMOS thresholds.

Although MOS transistors are symmetrical, by convention we say that majority carriers flow from their source to their drain. Because electrons are negatively charged, the source of an nMOS transistor is the more negative of the two terminals. Holes are positively charged so the source of a pMOS transistor is the more positive of the two terminals. In complementary CMOS gates, the source is the terminal closer to the supply rail and the drain is the terminal closer to the output.

The delay of MOS circuits is determined by the time required to charge or discharge the capacitance of the circuits. The gate of an MOS transistor is inherently a good capacitor with a thin dielectric; indeed, its capacitance is responsible for attracting carriers to the channel and thus for the operation of the device. The junctions of the reverse-biased p–n junctions from source or drain to the body contribute additional *parasitic* capacitance. The capacitance of wires interconnecting the transistors is also very significant and will be explored in Section 4.5.2.

We begin in Section 2.2 by deriving an idealized model relating current and voltage (I-V) for a transistor. This model provides a general understanding of transistor behavior but is of limited value. On the one hand, it neglects too many effects that are important in modern transistors with short channel lengths L. Therefore, the model is not sufficient to accurately calculate current. On the other hand, it is still too complicated to use in back-of-

[2]The terminology of source and drain might initially seem backward. Recall that the current in an nMOS transistor is carried by moving electrons with a negative charge. Therefore, positive current from drain to source corresponds to electrons flowing from their source to their drain.

the-envelope calculations when computing the performance of large circuits. Therefore, we will develop even simpler models for performance estimation in Section 2.6. To plausibly simulate transistor behavior, we must consider many second-order effects introduced in Section 2.4 and use circuit simulators based on SPICE [Nagel75] (see Chapter 5) with elaborate models capturing these effects. Nevertheless, the first-order models form a foundation for understanding the second-order effects. In particular, even the smallest transistors still act as voltage-controlled switches with a cutoff region in which the current from source to drain is nearly zero, a linear region in which the transistor acts as a resistor, and a saturation region in which current becomes nearly independent of drain-source voltage.

The remainder of this chapter develops quantitative models of MOS circuit operation. It begins with the ideal Shockley models of transistor I-V characteristics and a study of capacitances. It then explores nonidealities that are important for design. The models are used to predict the DC transfer characteristics of inverters and pass transistors circuits. Finally, a very simple resistor-capacitor (RC) model is introduced for delay estimation.

2.2 Ideal I-V Characteristics

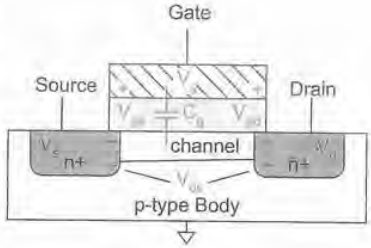
As stated previously, MOS transistors have three regions of operation:

- Cutoff or subthreshold region
- Linear or nonsaturation region
- Saturation region

Let us derive a first-order (ideal Shockley) model [Shockley52, Cobbold70, Sah64] relating the current and voltage (I-V) for an nMOS transistor in each of these regions. In the cutoff region ($V_{gs} < V_t$), there is no channel and almost zero current flows from drain to source. In the other regions, the gate attracts carriers (electrons) to form a channel. The electrons drift from source to drain at a rate proportional to the electric field between these regions. Thus we can compute currents if we know the amount of charge in the channel and the rate at which it moves. We know that the charge on each plate of a capacitor is $Q = CV$. Thus the charge in the channel $Q_{channel}$ is

$$Q_{channel} = C_g\left(V_{gc} - V_t\right) \tag{2.1}$$

where C_g is the capacitance of the gate to the channel and $V_{gc} - V_t$ is the amount of voltage attracting charge to the channel beyond the minimum required to invert from p to n. The gate voltage is referenced to the channel, which is not grounded. If the source is at V_s and the drain is at V_d, the average is $V_c = (V_s + V_d)/2 = V_s + V_{ds}/2$. Therefore, the mean difference between the gate and channel potentials V_{gc} is $V_{gs} - V_{ds}/2$, as shown in Figure 2.5.



Average gate to channel potential:

$V_{gc} = (V_{gs} + V_{gd})/2 = V_{gs} - V_{ds}/2$

FIG 2.5 Average gate to channel voltage

We can model the gate as a parallel plate capacitor with capacitance proportional to area over thickness. If the gate has length L and width W and the oxide thickness is t_{ox}, as shown in Figure 2.6, the capacitance is

$$C_g = \varepsilon_{ox} \frac{WL}{t_{ox}} \tag{2.2}$$

where the permittivity $\varepsilon_{ox} = 3.9\ \varepsilon_0$ for SiO_2 and ε_0 is the permittivity of free space, $8.85 \cdot 10^{-14}$ F/cm. Often the ε_{ox}/t_{ox} term is called C_{ox}, the capacitance per unit area of the gate oxide.

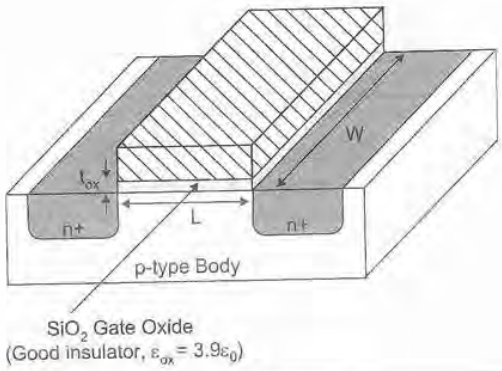


FIG 2.6 Transistor dimensions

Each carrier in the channel is accelerated to an average velocity proportional to the lateral electric field, i.e., the field between source and drain. The constant of proportionality μ is called the *mobility*.

$$v = \mu E \tag{2.3}$$

The electric field E is the voltage difference between drain and source V_{ds} divided by the channel length

$$E = \frac{V_{ds}}{L} \tag{2.4}$$

The time required for carriers to cross the channel is the channel length divided by the carrier velocity; L/v. Therefore, the current between source and drain is the total amount of charge in the channel divided by the time required to cross

$$\begin{aligned} I_{ds} &= \frac{Q_{\text{channel}}}{L/v} \\ &= \mu C_{ox} \frac{W}{L}\left(V_{gs} - V_t - V_{ds}/2\right)V_{ds} \\ &= \beta\left(V_{gs} - V_t - V_{ds}/2\right)V_{ds} \end{aligned} \tag{2.5}$$

where

$$\beta = \mu C_{ox} \frac{W}{L} \tag{2.6}$$

EQ (2.5) describes the linear region of operation, for $V_{gs} > V_t$, but V_{ds} relatively small. It is called *linear* or *resistive* because $V_{ds}/2 << V_{gs} - V_t$, I_{ds} increases almost linearly with V_{ds}, just like an ideal resistor. The geometry and technology-dependent parameters are sometimes merged into a single factor β. Do not confuse this use of β with the same symbol used for the ratio of collector-to-base current in a bipolar transistor. Some texts [Gray01] lump the technology-dependent parameters alone into a constant called "k prime."[3]

$$k' = \mu C_{ox} \tag{2.7}$$

However, if $V_{ds} > V_{dsat} \equiv V_{gs} - V_t$, the channel is no longer inverted in the vicinity of the drain; we say it is pinched off. Beyond this point, called the *drain saturation voltage*, increasing the drain voltage has no further effect on current. Substituting $V_{ds} = V_{dsat}$ at this point of maximum current into EQ (2.5), we find an expression for the saturation current that is independent of V_{ds}. This expression is valid for $V_{gs} > V_t$ and $V_{ds} > V_{dsat}$.

$$I_{ds} = \frac{\beta}{2}\left(V_{gs} - V_t\right)^2 \tag{2.8}$$

It is sometimes convenient to define I_{dsat} as the current of a transistor that is fully ON, i.e., $V_{gs} = V_{ds} = V_{DD}$.

$$I_{dsat} = \frac{\beta}{2}\left(V_{DD} - V_t\right)^2 \tag{2.9}$$

[3]Other sources (e.g., MOSIS) define $k' = \frac{\mu C_{ox}}{2}$; check the definition before using quoted data.

EQ (2.10) summarizes the current in the three regions:

$$I_{ds} = \begin{cases} 0 & V_{gs} < V_t \quad \text{cutoff} \\ \beta\left(V_{gs} - V_t - \frac{V_{ds}}{2}\right)V_{ds} & V_{ds} < V_{dsat} \quad \text{linear} \\ \frac{\beta}{2}\left(V_{gs} - V_t\right)^2 & V_{ds} > V_{dsat} \quad \text{saturation} \end{cases} \tag{2.10}$$

Example

Consider an nMOS transistor in a 180 nm process with $W/L = 4/2\ \lambda$ (i.e., 0.36/0.18 μm). In this process, the gate oxide thickness is 40 Å and the mobility of electrons is 180 cm²/V • *s* at 70° C. The threshold voltage is 0.4V. Plot I_{ds} vs. V_{ds} for V_{gs} = 0, 0.3, 0.6, 0.9, 1.2, 1.5, and 1.8 V.

Solution: We first calculate β.

$$\beta = \mu C_{ox}\frac{W}{L} = \left(180\ \frac{\text{cm}^2}{\text{V}\cdot\text{s}}\right)\left(\frac{3.9 \cdot 8.85 \cdot 10^{-14}\ \frac{\text{F}}{\text{cm}}}{40 \cdot 10^{-8}\ \text{cm}}\right)\left(\frac{W}{L}\right) = 155\frac{W}{L}\ \frac{\mu\text{A}}{\text{V}^2} \tag{2.11}$$

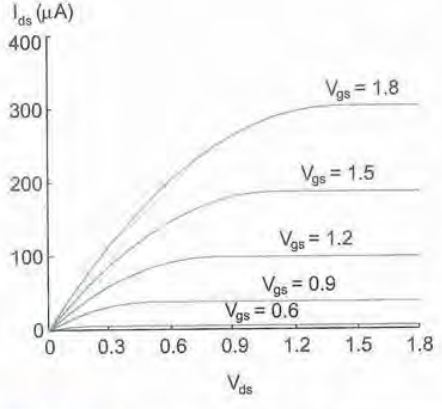


FIG 2.7 I-V characteristics of ideal nMOS transistor

Figure 2.7 shows the I-V characteristics for the transistor. According to the first-order model, the current is zero for gate voltages below V_t. For higher gate voltages, current increases linearly with V_{ds} for small V_{ds}. As V_{ds} reaches the saturation point $V_{gs} - V_t$, current rolls off and eventually becomes independent of V_{ds} when the transistor is saturated.

pMOS transistors behave in much the same way, but with the signs reversed and I-V characteristics in the third quadrant, as shown in Figure 2.8. The mobility of holes in silicon is typically lower than that of electrons. This means that pMOS transistors provide less current than nMOS transistors of comparable size and hence are slower. The symbols μ_n and μ_p are used to distinguish mobility of electrons and of holes in nMOS and pMOS transistors, respectively. The mobility ratio $\mu = \mu_n / \mu_p$ is typically 2–3; we will generally use 2 for examples in this book. The figure reflects a transistor of the same geometry as in Figure 2.7, but with μ_p = 90 cm²/V • s and $V_{tp} = -0.4$ V. Similarly, β_n, β_p, k'_n, and k'_p are sometimes used to distinguish nMOS and pMOS I-V characteristics.

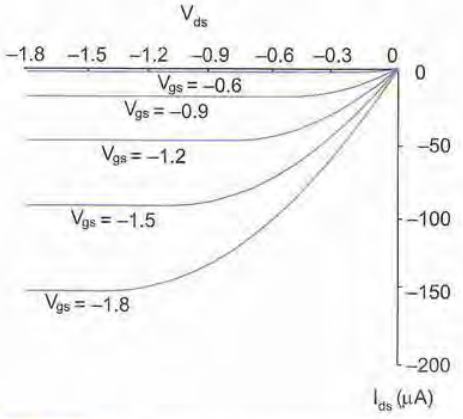


FIG 2.8 I-V characteristics of ideal pMOS transistor

2.3 C-V Characteristics

Each terminal of an MOS transistor has capacitance to the other terminals. In general, these capacitances are nonlinear and voltage dependent (C-V); however, they can be approximated as simple capacitors when their behavior is averaged across the switching voltages of a logic gate. This section first presents simple models of each capacitance suitable for estimating delay and power consumption of transistors. It then explores more detailed models used for circuit simulation. The more detailed models may be skipped on a first reading.

2.3.1 Simple MOS Capacitance Models

The gate of an MOS transistor is a good capacitor. Indeed, its capacitance is necessary to attract charge to invert the channel, so high gate capacitance is required to obtain high I_{ds}. As seen in Section 2.2, the gate capacitor can be viewed as a parallel plate capacitor with the gate on top and channel on bottom with the thin oxide dielectric between. Therefore, the capacitance is

$$C_g = C_{ox}WL \tag{2.12}$$

A capacitor is a two-terminal device. When the transistor is on, the channel extends from the source (and reaches the drain if the transistor is unsaturated, or stops short in saturation). It is not an unreasonable simplification to approximate the gate capacitance as terminating at the source and thus call the capacitance C_{gs}.

Most transistors used in logic are of minimum manufacturable length because this results in greatest speed and lowest power consumption. Thus taking this minimum L as a constant for a particular process, we can define

$$C_g = C_{\text{permicron}} \cdot W \tag{2.13}$$

where

$$C_{\text{permicron}} = C_{\text{ox}} L = \frac{\varepsilon_{\text{ox}}}{t_{\text{ox}}} L \tag{2.14}$$

Notice that if we develop a more advanced manufacturing process in which both the channel length and oxide thickness are reduced by the same factor, $C_{\text{permicron}}$ remains unchanged (and has a value of about 1.5–2 fF/μm of gate width). Table 5.5 lists gate capacitance for a variety of processes.

In addition to the gate, the source and drain also have capacitances. These capacitances are not fundamental to operation of the devices, but do impact circuit performance and hence are called *parasitic* capacitors. They arise from the reverse-biased p–n junctions between the source or drain diffusion and the body and hence are also called *diffusion*[4] capacitance C_{sb} and C_{db}. The size of these junctions depends on the area and perimeter of the source and drain diffusion, the depth of the diffusion, the doping levels, and the voltage. As diffusion has both high capacitance and high resistance, it is generally made as small as possible in the layout. There are three types of diffusion regions frequently seen, illustrated with the two series transistors in Figure 2.9. In Figure 2.9(a), each source and drain has its own *isolated* region of contacted diffusion. In Figure 2.9(b), the drain of the bottom transistor and source of the top transistor form a *shared* contacted diffusion region. In Figure 2.9(c), the source and drain are *merged* into an uncontacted region. The average capacitance of each of these types of regions can be calculated or measured from simulation as a transistor switches between V_{DD} and GND. Table 5.5 also lists the capacitance for each scenario for a variety of processes.

For the purposes of hand estimation, you can observe that the diffusion capacitance C_{sb} and C_{db} of contacted source and drain regions is comparable to the gate capacitance (e.g., 1.5-2 fF/μm of gate width). The diffusion capacitance of the uncontacted source or drain is somewhat less because the area is smaller but the difference is usually unimportant for hand calculations. These values of $C_g = C_{sb} = C_{db} \sim 2$fF/μm will be used in examples throughout the text, but you should obtain the appropriate data for your process using methods to be discussed in Section 5.4.

[4]Device engineers more properly call this *depletion* capacitance, but the term *diffusion* capacitance is widely used by circuit designers.

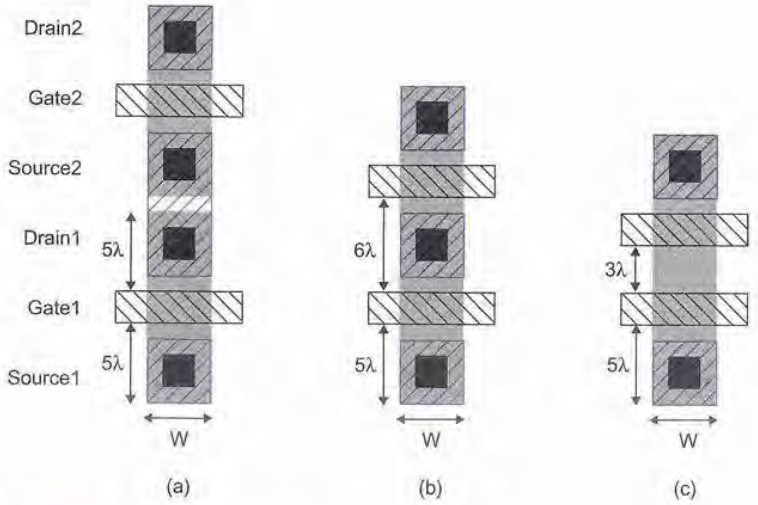


FIG 2.9 Diffusion region geometries

2.3.2 Detailed MOS Gate Capacitance Model



The MOS gate sits above the channel and may partially overlap the source and drain diffusion areas. Therefore, the gate capacitance has two components: the intrinsic capacitance (over the channel) and the overlap capacitances (to the source, drain, and body).

The intrinsic capacitance was approximated as a simple parallel plate in EQ (2.12). Let us call this capacitance $C_0 = WLC_{ox}$. However, the bottom plate of the capacitor depends on the mode of operation of the transistor.

1. *Cutoff.* When the transistor is OFF ($V_{gs} = 0$), the channel is not inverted and charge on the gate is matched with opposite charge from the body. This is called C_{gb}, the gate-to-body capacitance. As V_{gs} increases but remains below a threshold, a depletion region forms at the surface. This effectively moves the bottom plate downward from the oxide, reducing the capacitance.
2. *Linear.* When $V_{gs} > V_t$, the channel inverts and again serves as a good conductive bottom plate. However, the channel is connected to the source and drain, rather than the body. At low values of V_{ds}, the channel charge is roughly shared between source and drain, so $C_{gs} = C_{gd} = C_0/2$. As V_{ds} increases, the region near the drain becomes less inverted, so a greater fraction of the capacitance is attributed to the source and a smaller fraction to the drain.

3. *Saturation.* At $V_{ds} > V_{gs} - V_t$, the transistor saturates and the channel pinches off. At this point, all the intrinsic capacitance is to the source. Because of pinchoff, the capacitance in saturation reduces to $C_{gs} = 2/3\ C_0$ for an ideal transistor [Gray01].

The behavior in these three regions can be approximated as shown in Table2.1.

Table 2.1 Approximation of intrinsic MOS gate capacitance

Parameter	Cutoff	Linear	Saturation
C_{gb}	C_0	0	0
C_{gs}	0	$C_0/2$	$2/3\ C_0$
C_{gd}	0	$C_0/2$	0
$C_g = C_{gs} + C_{gd} + C_{gb}$	C_0	C_0	$2/3\ C_0$

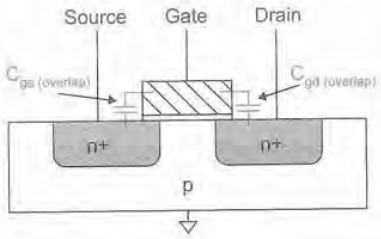
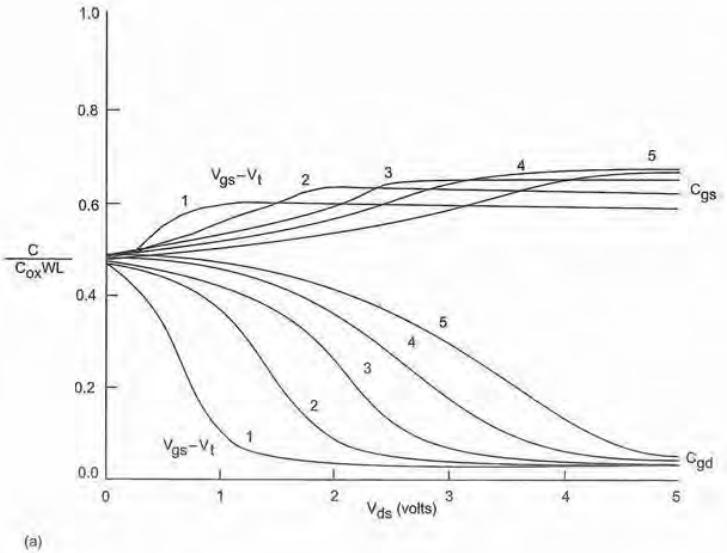


FIG 2.10 Overlap capacitance

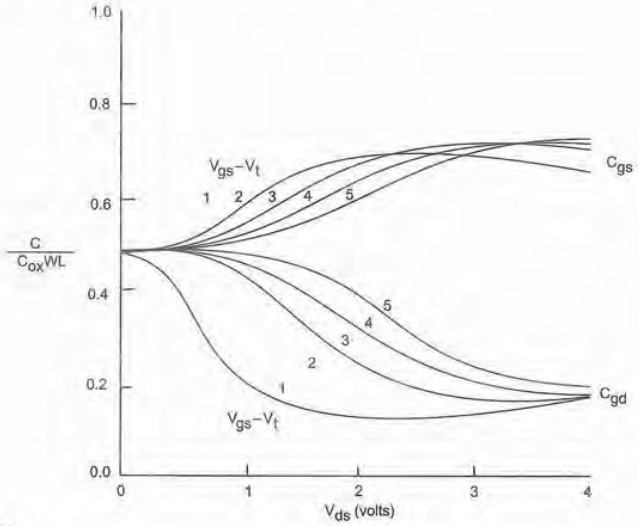
The gate overlaps the source and drain by a small amount in a real device and also has fringing fields terminating on the source and drain. This leads to additional overlap capacitances, as shown in Figure 2.10. These capacitances are proportional to the width of the transistor. Typical values are $C_{gsol} = C_{gdol} =$ 0.2–0.4 fF/μm.

$$C_{gs(\text{overlap})} = C_{gsol}W$$
$$C_{gd(\text{overlap})} = C_{gdol}W \qquad (2.15)$$

The experimentally measured C_{gs} and C_{gd} of a long channel nMOS transistor (W = 49.2 μm, L = 4.5 μm) is shown in Figure 2.11(a) [Sheu87a]. This graph shows the normalized capacitance varying as a function of V_{ds} for a number of $V_{gs} - V_t$ values. Observe that at $V_{ds} = 0$, $C_{gs} = C_{gd} = C_0/2$. As V_{ds} increases, the capacitances approach $C_{gs} = 2/3\ C_0$ and $C_{gd} = 0$, as expected when the transistor is saturated. Figure 2.11(b) shows measured capacitances of a shorter channel transistor (W = 49.2 μm, L = 0.75 μm). Observe that C_{gd} does not go to 0 in saturation because the overlap component $C_{gd(\text{overlap})}$ is significant. Overlap capacitance becomes relatively more important for shorter channel transistors because it is a larger fraction of the total.



(a)



(b)

FIG 2.11 Total gate capacitance of an MOS transistor as a function of V_{ds}. ©IEEE 1987.

It is convenient to view the gate capacitance as a single-terminal capacitor attached to the gate. Because the source and drain actually form second terminals, the effective gate capacitance varies with the switching activity of the source and drain. Figure 2.12 shows the effective gate capacitance in a 0.35 μm process for seven different combinations of source and drain behavior [Bailey98].

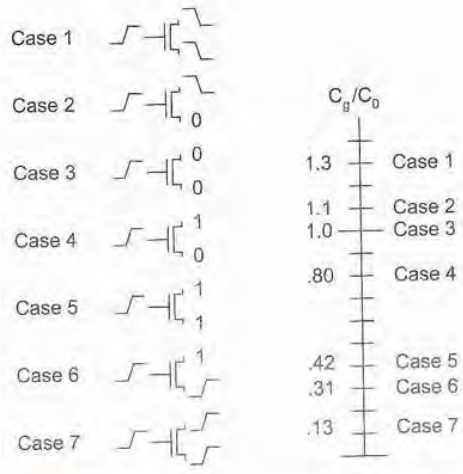


FIG 2.12 Data-dependent gate capacitance

More accurate modeling of the gate capacitance may be achieved by using a charge-based model [Cheng99]. The overlap capacitance also displays a voltage dependence. For the purpose of delay calculation of digital circuits, we usually approximate $C_g = C_{gs} + C_{gd} + C_{gb} \sim C_0$.

2.3.3 Detailed MOS Diffusion Capacitance Model

OPTIONAL

As mentioned in Section 2.3.1, the reverse-biased p–n junction between the source diffusion and the body contributes parasitic capacitance. The capacitance depends on both the *area AS* and *sidewall perimeter PS* of the source diffusion region. The geometry is illustrated in Figure 2.13. The area is $AS = W \cdot D$. The perimeter is $PS = 2 \cdot W + 2 \cdot D$. Of this perimeter, W abuts the gate and the remaining $W + 2 \cdot D$ does not.

The total source parasitic capacitance is

$$C_{sb} = AS \cdot C_{jbs} + PS \cdot C_{jbssw} \tag{2.16}$$

where C_{jbs} has units of capacitance/area and C_{jbssw} has units of capacitance/length.

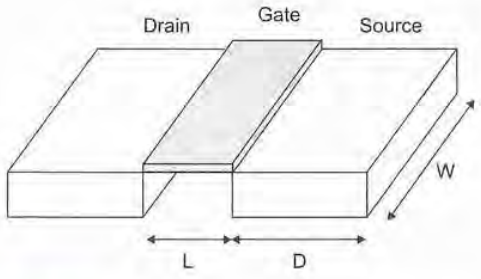


FIG 2.13 Diffusion region geometry

Because the depletion region thickness depends on the reverse bias, these parasitics are nonlinear. The area junction capacitance term is

$$C_{jbs} = C_J \left(1 + \frac{V_{sb}}{\psi_0}\right)^{-M_J} \tag{2.17}$$

C_J is the junction capacitance at zero bias and is highly process-dependent. M_J is the *junction grading coefficient*, typically in the range of 0.5 to 0.33 depending on the abruptness of the diffusion junction. ψ_0 is the *built-in potential* that depends on doping levels.

$$\psi_0 = v_T \ln \frac{N_A N_D}{n_i^2} \tag{2.18}$$

v_T is the *thermal voltage* from thermodynamics, not to be confused with the threshold voltage V_t. It has a value equal to kT/q (26 mV at room temperature), where $k = 1.380 \cdot 10^{-23}$ J/K is Boltzmann's constant, T is absolute temperature (300 K at room temperature), and $q = 1.602 \cdot 10^{-19}$ C is the charge of an electron. N_A and N_D are the doping levels of the body and source diffusion region. n_i is the intrinsic carrier concentration in undoped silicon and has a value of $1.45 \cdot 10^{10}$ cm^{-3} at 300 K.

The sidewall capacitance term is of a similar form but uses different coefficients.

$$C_{jbssw} = C_{JSW} \left(1 + \frac{V_{sb}}{\psi_0}\right)^{-M_{JSW}} \tag{2.19}$$

The capacitance contributed by the sidewall facing the channel can be modified somewhat by the presence of the channel depletion region and the modified doping profiles. In some

SPICE models, the capacitance of this sidewall abutting the gate is specified with another set of parameters:

$$C_{jbswg} = C_{JSWG}\left(1+\frac{V_{sb}}{\Psi_0}\right)^{-M_{JSWG}} \tag{2.20}$$

Section 5.3.4 discusses SPICE perimeter capacitance models further.

The drain diffusion has a similar parasitic capacitance dependent on *AD*, *PD*, and V_{db}. Equivalent relationships hold for pMOS transistors, but doping levels differ. As the capacitances are voltage-dependent, the most useful information to digital designers is the value averaged across a switching transition. This is the C_{sb} or C_{db} value that was presented in Section 2.3.1. Analog designers must minimize the consequences of these variations by using good circuit design.

Example

Calculate the diffusion parasitic C_{db} of the drain of a unit-sized contacted nMOS transistor in a 180 nm process when the drain is at 0 and at V_{DD} = 1.8 V. Assume the substrate is grounded. The transistor characteristics are *CJ* = 0.98 fF/μm², *MJ* = 0.36, *CJSW* = 0.22 fF/μm, *CJSWG* = 0.33 fF/μm, *MJSW* = *MJSWG* = 0.10, and ψ_0 = 0.75 V at room temperature.

Solution: From Figure 2.9 we find a unit-size diffusion contact is 4 x 5 λ, or 0.36 x 0.45 μm. The area is 0.162 μm² and perimeter is 1.26 μm plus 0.36 μm along the gate. At zero bias, C_{jbs} = 0.98 fF/μm², C_{jbssw} = 0.22 fF/μm, and C_{jbswg} = 0.33 fF/μm. Hence the total capacitance is

$$\begin{aligned} C_{db}(0\text{ V}) &= \left(0.162\ \mu\text{m}^2\right)\left(0.98\ \tfrac{\text{fF}}{\mu\text{m}^2}\right)+ \\ &\left(1.26\ \mu\text{m}\right)\left(0.22\ \tfrac{\text{fF}}{\mu\text{m}}\right)+\left(0.36\ \mu\text{m}\right)\left(0.33\ \tfrac{\text{fF}}{\mu\text{m}}\right)=0.55\ \text{fF} \end{aligned} \tag{2.21}$$

At a drain voltage of V_{DD}, the capacitance reduces to

$$\begin{aligned} C_{db}(1.8\text{ V}) &= \left(0.162\ \mu\text{m}^2\right)\left(0.98\ \tfrac{\text{fF}}{\mu\text{m}^2}\right)\left(1+\frac{1.8}{0.75}\right)^{-0.36}+ \\ &\left[\left(1.26\ \mu\text{m}\right)\left(0.22\ \tfrac{\text{fF}}{\mu\text{m}}\right)+\left(0.36\ \mu\text{m}\right)\left(0.33\ \tfrac{\text{fF}}{\mu\text{m}}\right)\right]\left(1+\frac{1.8}{0.75}\right)^{-0.10}=0.44\ \text{fF} \end{aligned} \tag{2.22}$$

For the purpose of manual performance estimation, this nonlinear capacitance is too much effort. An effective capacitance averaged over the switching range is quite satisfactory for digital applications.

Diffusion regions were historically used for short wires called *runners* in processes with limited numbers of metal levels. Diffusion capacitance and resistance are large enough that such practice is now discouraged.

In summary, an MOS transistor can be viewed as a four-terminal device with capacitances between each terminal pair as shown in Figure 2.14. The gate capacitance includes an intrinsic component (to the body, source and drain, or source alone, depending on operating regime) and overlap terms with the source and drain. The source and drain have parasitic diffusion capacitance to the body.

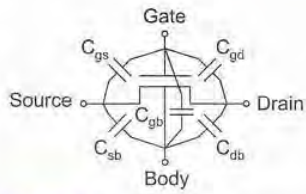


FIG 2.14 Capacitances of an MOS transistor

2.4 Nonideal I-V Effects

The ideal I-V model of EQs (2.5) and (2.8) neglects many effects that are important to modern devices. Figure 2.15 shows the simulated I-V characteristics of a unit nMOS transistor in a 180 nm process.

Compare the characteristics in the linear and saturation regimes (Figure 2.15(a)) to those of the ideal device in Figure 2.7. The saturation current increases less than quadratically with increasing V_{gs}. This is caused by two effects: *velocity saturation* and *mobility degradation*. At high lateral field strengths (V_{ds}/L), carrier velocity ceases to increase linearly with field strength. This is called velocity saturation and results in lower I_{ds} than expected at high V_{ds}. At high vertical field strengths (V_{gs}/t_{ox}), the carriers scatter more often. This mobility degradation effect also leads to less current than expected at high V_{gs}. The saturation current of the nonideal transistor increases slightly with V_{ds}. This is caused by *channel length modulation*, in which higher V_{ds} increases the size of the depletion region around the drain and thus effectively shortens the channel.

There are several sources of leakage resulting in current flow in nominally OFF transistors. Observe in Figure 2.15(b) that at $V_{gs} < V_t$, the current drops off exponentially rather than abruptly becoming zero. This is called *subthreshold conduction*. The threshold voltage itself is influenced by the voltage difference between the source and body; this is called the body effect. The source and drain diffusions are reverse-biased diodes and also experience *junction leakage* into the substrate or well. The current into the gate I_g is ideally 0. However, as the thickness of gate oxides reduces to only a small number of atomic layers, electrons *tunnel* through the gate, causing some gate current.

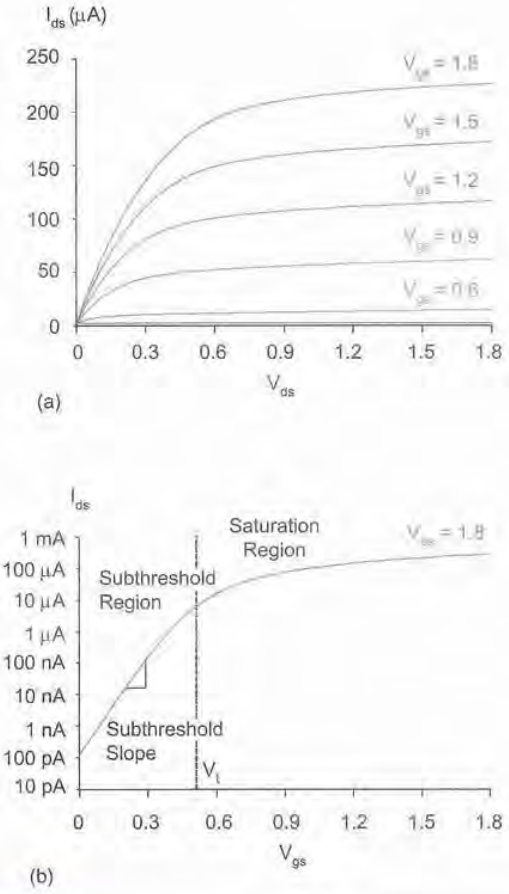


FIG 2.15 Simulated I-V characteristics

Both mobility and threshold voltage decrease with rising temperature. The mobility effect is most important for ON transistors, resulting in lower I_{ds} at high temperature. The threshold effect is most important for OFF transistors, resulting in higher leakage current at high temperature. Clearly, MOS characteristics degrade with temperature.

It is useful to have a qualitative understanding of nonideal effects to predict their impact on circuit behavior and to be able to anticipate how devices will change in future process generations. However, the effects lead to complicated I-V characteristics that are seldom useful for hand calculations. Instead, the effects are built into good transistor models and simulated with SPICE or similar software.

2.4.1 Velocity Saturation and Mobility Degradation

According to EQ (2.3), carrier drift velocity and hence current increase linearly with the lateral electric field $E_{lat} = V_{ds}/L$ between source and drain. This is only true for weak fields; at high field strength, drift velocity rolls off due to carrier scattering and eventually saturates[5] at v_{sat}, as shown in Figure 2.16. The carrier velocity may be fitted with EQ (2.23) [Murphy80, Cheng99] where E_{sat} is determined empirically. $v_{sat} = \mu E_{sat}$ is in the range of $6-10 \cdot 10^6$ cm/s for electrons and $4-8 \cdot 10^6$ cm/s for holes. This corresponds to a saturation field on the order of $2 \cdot 10^4$ V/cm for nMOS transistors.

$$v = \frac{\mu E_{lat}}{1 + \frac{E_{lat}}{E_{sat}}} \tag{2.23}$$

Recall that without velocity saturation, the saturation current is

$$I_{ds} = \mu C_{ox} \frac{W}{L} \frac{(V_{gs} - V_t)^2}{2} \tag{2.24}$$

If the transistor were completely velocity saturated, $v = v_{sat}$ and the saturation current becomes [Bakoglu90]

$$I_{ds} = C_{ox} W (V_{gs} - V_t) v_{sat} \tag{2.25}$$

Observe that the drain current is quadratically dependent on voltage without velocity saturation and linearly dependent when fully velocity saturated. For moderate supply voltages, transistors operate in a region where the velocity no longer increases linearly with field, but also is not completely saturated. The *α-power law model* given in EQ (2.26) provides a simple approximation to capture this behavior [Sakurai90]. α is called the *velocity saturation index* and is determined by curve fitting measured I-V data. Transistors with long channels or low V_{DD} display quadratic I-V characteristics in saturation and are mod-

[5] Do not confuse the *saturation* region of transistor operation (where $V_{ds} > V_{gs} - V_t$) with *velocity saturation* (where $E_{lat} = V_{ds}/L$ approaches E_{sat}). In this text, the word "saturation" alone refers to the operating region while "velocity saturation" refers to the limiting of carrier velocity at high field.

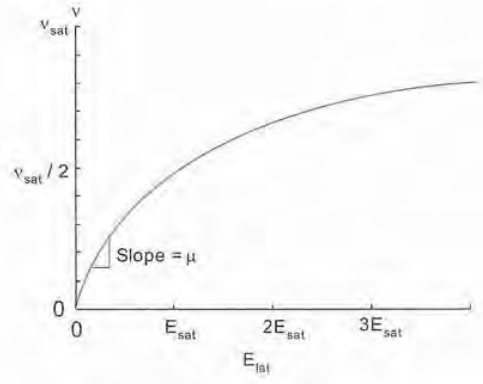


FIG 2.16 Carrier velocity vs. electric field

eled with α = 2. As transistors become more velocity saturated, increasing V_{gs} has less effect on current and α decreases, reaching 1 for transistors that are completely velocity saturated. For simplicity, the model uses a straight line in the linear region. Overall, the model is based on three parameters that can be determined empirically from a curve fit of I-V characteristics: α, βP_c, and P_v.

$$I_{ds} = \begin{cases} 0 & V_{gs} < V_t \quad \text{cutoff} \\ I_{dsat}\dfrac{V_{ds}}{V_{dsat}} & V_{ds} < V_{dsat} \quad \text{linear} \\ I_{dsat} & V_{ds} > V_{dsat} \quad \text{saturation} \end{cases} \tag{2.26}$$

where

$$\begin{aligned} I_{dsat} &= P_c \frac{\beta}{2}\left(V_{gs} - V_t\right)^{\alpha} \\ V_{dsat} &= P_v\left(V_{gs} - V_t\right)^{\alpha/2} \end{aligned} \tag{2.27}$$

As channel lengths become shorter, the lateral field increases and transistors become more velocity saturated (α closer to 1) if the supply voltage is held constant. For example, a transistor with a 2 μm channel length begins to show the effects of velocity saturation at V_{DD} above 4 V, while a 0.18 μm long transistor begins to experience velocity saturation above 0.36 V. Figure 2.17 compares I_{ds} for a velocity-saturated nMOS transistor with that of an ideal (Shockley model) transistor from Figure 2.17 and with that predicted by the α-power law. The Shockley model grossly overpredicts current at high voltage but the α-power fit is reasonably good. As the transistor becomes severely velocity saturated, there is no performance benefit to raising V_{DD}.

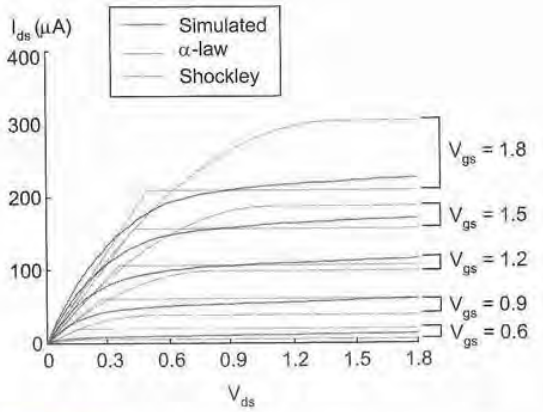


FIG 2.17 I-V characteristics for nMOS transistor with velocity saturation

The low-field mobility of holes is much lower than that of electrons, so pMOS transistors experience less velocity saturation than nMOS for a given V_{DD}. This shows up as a larger value of α for pMOS than for nMOS transistors.

Strong vertical electric fields resulting from large V_{gt} cause the carriers to scatter against the surface and also reduce the carrier mobility μ. This effect is called *mobility degradation*. It can be modeled by replacing μ with a smaller μ_{eff}. The α-power law captures this effect in the parameter α.

2.4.2 Channel Length Modulation

Ideally, I_{ds} is independent of V_{ds} for a transistor in saturation, making the transistor a perfect current source. As discussed in Section 2.3.3, the reverse-biased p–n junction between the drain and body forms a depletion region with a width L_d that increases with V_{db}. The depletion region effectively shortens the channel length to

$$L_{eff} = L - L_d \tag{2.28}$$

To avoid introducing the body voltage into our calculations, assume the source voltage is close to the body voltage so $V_{db} \sim V_{ds}$. Hence, increasing V_{ds} decreases the effective channel length. Shorter channel length results in higher current; thus I_{ds} increases with V_{ds} in saturation as shown in Figure 2.18. This can be crudely modeled by multiplying EQ (2.10) by a factor of $(1 + \lambda V_{ds})$ [Gray01]. In the saturation region, we find

$$I_{ds} = \beta \frac{(V_{gs} - V_t)^2}{2}(1 + \lambda V_{ds}) \tag{2.29}$$

The parameter λ is an empirical channel length modulation factor that should not be confused with the same symbol used in layout design rules. As channel length gets shorter, the effect of the channel length modulation becomes relatively more important. Hence λ is inversely dependent on channel length. This λ channel length modulation model is a gross oversimplification of nonlinear behavior and is more useful for conceptual understanding than for accurate device modeling.

Channel length modulation is very important to analog designers because it reduces the gain of amplifiers. It is generally unimportant for qualitatively understanding the behavior of digital circuits.

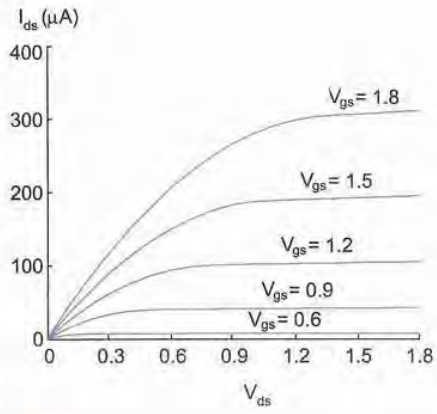


FIG 2.18 I-V characteristics of nMOS transistor with channel length modulation

2.4.3 Body Effect

Until now, we have considered a transistor to be a three-terminal device with gate, source, and drain. However, the body is an implicit fourth terminal. The potential difference between the source and body V_{sb} affects the threshold voltage. The threshold voltage can be modeled as

$$V_t = V_{t0} + \gamma\left(\sqrt{\phi_s + V_{sb}} - \sqrt{\phi_s}\right) \qquad (2.30)$$

where V_{t0} is the threshold voltage when the source is at the body potential, ϕ_s is the *surface potential* at threshold (see a device physics text for further discussion of surface potential), and γ is the *body effect coefficient*, typically in the range 0.4 to 1 $V^{1/2}$. These in turn depend on the doping level N_A. Section 4.4.3.2 will describe how a body bias can intentionally be applied to alter the threshold voltage, permitting tradeoffs between performance and subthreshold leakage current.

$$\phi_s = 2v_T \ln\frac{N_A}{n_i} \qquad (2.31)$$

$$\gamma = \frac{t_{ox}}{\varepsilon_{ox}}\sqrt{2q\varepsilon_{si}N_A} = \frac{\sqrt{2q\varepsilon_{si}N_A}}{C_{ox}} \qquad (2.32)$$

Example

Consider the nMOS transistor in a 180 nm process with a nominal threshold voltage of 0.4 V and doping level of $8 \cdot 10^{17}$ cm^{-3}. The body is tied to ground with a substrate contact. How much does the threshold change at room temperature if the source is at 1.1 V instead of 0?

Solution: At room temperature the thermal voltage $v_T = kT/q = 26$ mV and $n_i = 1.45 \cdot 10^{10}$ cm^{-3}. The threshold increases by 0.28 V.

$$\phi_s = 2(0.026\text{ V})\ln\frac{8 \cdot 10^{17}\text{ cm}^{-3}}{1.45 \cdot 10^{10}\text{ cm}^{-3}} = 0.93\text{ V}$$

$$\gamma = \frac{40 \cdot 10^{-8}\text{ cm}}{3.9 \cdot 8.85 \cdot 10^{-14}\ \frac{\text{F}}{\text{cm}}}\sqrt{2\left(1.6 \cdot 10^{-19}\text{ C}\right)\left(11.7 \cdot 8.85 \cdot 10^{-14}\ \frac{\text{F}}{\text{cm}}\right)\left(8 \cdot 10^{17}\text{ cm}^{-3}\right)} = 0.60\text{ V}^{\frac{1}{2}} \quad (2.33)$$

$$V_t = 0.4\text{ V} + \gamma\left(\sqrt{\phi_s + 1.1\text{ V}} - \sqrt{\phi_s}\right) = 0.68\text{ V}$$

2.4.4 Subthreshold Conduction

The ideal transistor I-V model assumes current only flows from source to drain when $V_{gs} > V_t$. In real transistors, current does not abruptly cut off below threshold, but rather drops off exponentially as given in EQ(2.34) [Sheu87b, Cheng99]. This conduction is also known as *leakage* and often results in undesired current when a transistor is nominally OFF. I_{ds0} is the current at threshold and is dependent on process and device geometry; the $e^{1.8}$ term was found empirically. n is a process-dependent term affected by the depletion region characteristics and is typically in the range of 1.4–1.5 for CMOS processes. The final term indicates that leakage is 0 if $V_{ds} = 0$, but increases to its full value when V_{ds} is a few multiples of the thermal voltage v_T (e.g., when $V_{ds} > 50$ mV). Figure 2.15(b) shows the I-V characteristics on a logarithmic scale illustrating both normal and subthreshold conduction.

$$I_{ds} = I_{ds0}e^{\frac{V_{gs}-V_t}{nv_T}}\left(1 - e^{\frac{-V_{ds}}{v_T}}\right) \quad (2.34)$$

$$I_{ds0} = \beta v_T^2 e^{1.8} \quad (2.35)$$

Subthreshold conduction is used to advantage in very low-power analog circuits. It is also particularly important for dynamic circuits and DRAMs, which depend on the storage of charge on a capacitor. Conduction through an OFF transistor discharges the capacitor unless it is periodically refreshed or a trickle of current is available to counter the leakage. Leakage also contributes to power dissipation in idle circuits. Leakage increases exponentially as V_t decreases or as temperature rises, so it is becoming a major problem for chips using low supply and threshold voltages.

Subthreshold conduction is exacerbated by *drain-induced barrier lowering* (DIBL) in which a positive V_{ds} effectively reduces V_t. This effect is especially pronounced in short-channel transistors. It can be modeled as

$$V_t' = V_t - \eta V_{ds} \quad (2.36)$$

Example

What is the minimum threshold voltage for which the leakage current through an OFF transistor ($V_{gs} = 0$) is 10^3 times less than that of a transistor that is barely ON ($V_{gs} = V_t$) at room temperature if $n = 1.5$? One of the advantages of silicon-on-insulator (SOI) processes is that they have smaller n (see Section 6.7). What threshold is required for SOI if $n = 1.3$?

Solution: $v_T = 26$ mV at room temperature. Assume $V_{ds} >> v_T$ so leakage is significant. We solve

$$I_{ds}(V_{gs} = 0) = 10^{-3} I_{ds0} \approx I_{ds0} e^{\frac{-V_t}{nv_T}} \tag{2.37}$$

$$V_t = -nv_T \ln 10^{-3} = 270 \text{ mV}$$

In the CMOS process, leakage rolls off by a factor of 10 for every 90 mV V_{gs} falls below threshold. This is often quoted as a *subthreshold slope* of $S = 90$ mV/decade. In the SOI process, the subthreshold slope S is 78 mV/decade, so a threshold of only 234 mV is required.

where η is the DIBL coefficient, typically in the range of 0.02–0.1.

[Bowman99] extends the α-power law model to a *physical α-power law model* that includes subthreshold conduction as well as velocity saturation effects.

2.4.5 Junction Leakage

The p–n junctions between diffusion and the substrate or well form diodes, as shown in Figure 2.19. The well-to-substrate junction is another diode. The substrate and well are tied to GND or V_{DD} to ensure these diodes remain reverse-biased. However, reverse-biased diodes still conduct a small amount of current I_D.[6]

$$I_D = I_S \left(e^{\frac{V_D}{v_T}} - 1 \right) \tag{2.38}$$

where I_S depends on doping levels and on the area and perimeter of the diffusion region and V_d is the diode voltage (e.g., V_{sb} or V_{db}). When a junction is reverse biased by significantly more than the thermal voltage, the leakage is just $-I_S$, generally in the 0.1–0.01 fA/μm^2 range.

[6]Beware that I_D and I_S stand for the diode current and diode reverse-biased saturation currents, respectively. The D and S are not related to drain or source.

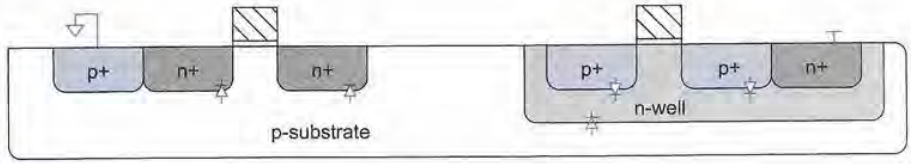


FIG 2.19 Reverse-biased diodes in CMOS circuits

Junction leakage was historically a limiter of storage time on dynamic nodes. In modern transistors with low threshold voltages, subthreshold conduction far exceeds junction leakage.

2.4.6 Tunneling

According to quantum mechanics, there is a finite probability that carriers will tunnel through the gate oxide. This results in *gate leakage* current flowing into the gate. The probability of tunneling drops off exponentially with oxide thickness, and so was negligible until recently.

For oxides thinner than about 15–20 Å, tunneling current becomes a factor and may become comparable to subthreshold leakage in advanced processes. Figure 2.20 plots gate leakage current density J_G against voltage for various oxide thicknesses. Large tunneling currents impact not only dynamic nodes but also quiescent power consumption and thus may limit oxide thicknesses t_{ox} to no less than about 8Å [Song01]. To keep dimensions in perspective, recall that each atomic layer of SiO_2 is about 3 Å, so gate oxides have scaled to only a handful of atomic layers thick. High C_{ox} is important for good transistors, so one research direction is to use an alternative gate insulator with a higher dielectric constant ϵ. A key challenge is finding materials that form a high-quality interface with silicon; one contender is silicon nitride (Si_3N_4) with a dielectric constant of 7.8. Tunneling can purposely be used to create electrically erasable memory devices (see Section 11.3).

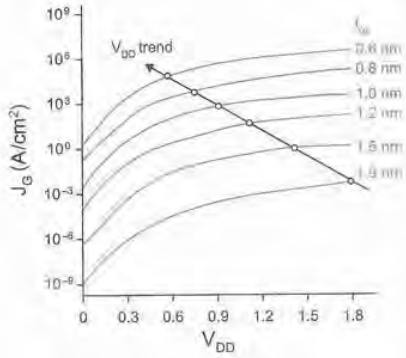


FIG 2.20 Gate leakage current from [Song01]

Tunneling current is an order of magnitude higher for nMOS than pMOS transistors with SiO_2 gate dielectrics because the electrons tunnel from the conduction band while the holes tunnel from the valence band and see a higher barrier [Hamzaoglu02]. Different dielectrics may have different tunneling properties.

2.4.7 Temperature Dependence

Transistor characteristics are influenced by temperature [Cobbold66, Vadasz66, Tsividis99, Gutierrez01]. Carrier mobility decreases with temperature. An approximate relation is

$$\mu(T) = \mu(T_r)\left(\frac{T}{T_r}\right)^{-k_\mu} \tag{2.39}$$

where T is the absolute temperature, T_r is room temperature, and k_μ is a fitting parameter generally in the range of 1.2–2.0.

The magnitude of the threshold voltage decreases nearly linearly with temperature and may be approximated by

$$V_t(T) = V_t(T_r) - k_{vt}(T - T_r) \tag{2.40}$$

where k_{vt} is typically in the range of 0.5 to 3.0 mV/K.

Junction leakage also increases with temperature because I_S is strongly temperature dependent.

The combined temperature effects are shown in Figure 2.21, where ON current decreases and OFF current increases with temperature. Similarly, Figure 2.22 shows how the ON current I_{dsat} decreases with temperature. Therefore, circuit performance is generally worst at high temperature. This is called a *negative temperature coefficient*.

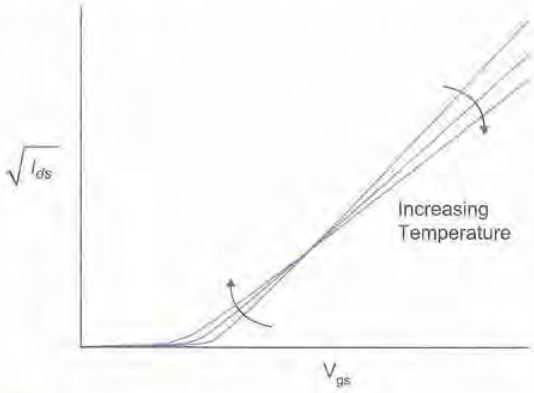


FIG 2.21 I–V characteristics of nMOS transistor in saturation at various temperatures

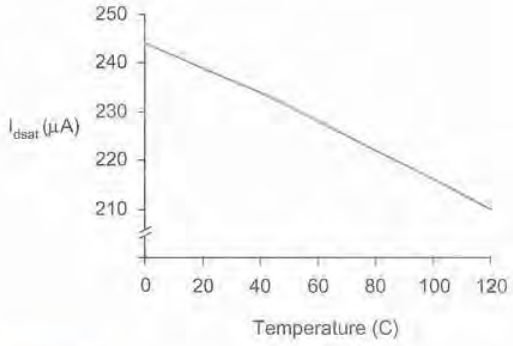


FIG 2.22 I_{dsat} vs. temperature

Conversely, circuit performance can be improved by cooling. Most systems use natural convection or fans in conjunction with heat sinks, but water cooling, thin-film refrigerators, or even liquid nitrogen can increase performance if the expense is justified. There are many advantages of operating at low temperature [Keyes70, Sun87]. Subthreshold leakage is exponentially dependent on temperature, so lower threshold voltages can be used. Velocity saturation occurs at higher fields, providing more current. As mobility is also higher, these fields are reached at a lower power supply, saving power. Depletion regions become wider, resulting in less junction capacitance. Most wearout mechanisms are tem-

perature-dependent, so transistors are more reliable. However, at low temperatures, transistors break down at lower voltages.

Two popular lab tools for determining temperature dependence in circuits are a can of freeze spray and a heat gun. The former can be used to momentarily "freeze" a chip to see whether performance alters and the other, of course, can be used to heat a chip up. Often these tests are done to quickly determine whether a chip is prone to temperature effects. Be careful—sometimes the sudden temperature change can fracture chips or their packages.

2.4.8 Geometry Dependence

The layout designer draws transistors with width and length W_{drawn} and L_{drawn}. The actual gate dimensions may differ by some factors X_W and X_L. For example, the manufacturer may create masks with narrower polysilicon or may overetch the polysilicon to provide shorter channels (negative X_L) without changing the overall design rules or metal pitch. Moreover, the source and drain tend to diffuse laterally under the gate by L_D, producing a shorter effective channel length that the carriers must traverse between source and drain. Similarly, diffusion of the bulk by W_D decreases the effective channel width. Putting these factors together, we can compute effective transistor lengths and widths that should be used in place of L and W in the current and capacitance equations given elsewhere in the book. The factors of two come from lateral diffusion on both sides of the channel.

$$\begin{aligned} L_{eff} &= L_{drawn} + X_L - 2L_D \\ W_{eff} &= W_{drawn} + X_W - 2W_D \end{aligned} \tag{2.41}$$

Therefore, a transistor drawn twice as long may have an effective length that is more than twice as great. Similarly, two transistors differing in drawn widths by a factor of two may differ in saturation current by more than a factor of two. Threshold voltages also vary somewhat with transistor dimensions [Cheng99].

Recall that long transistors also experience less channel length modulation. Combining the threshold, effective channel length, and channel length modulation effects, a transistor of twice minimum length usually delivers substantially less than half the current of a minimum length device. In general, when currents must be precisely matched (e.g., in sense amplifiers or A/D converters), it is best to use the same width and length for each device. Current ratios can be produced by tying several identical transistors in parallel.

In processes below 0.25 μm, the effective length of the transistor also depends significantly on the orientation of the transistor. Moreover, the amount of nearby polysilicon also affects etch rates during manufacturing and thus channel length. Transistors that must match well should have the same orientation. Dummy polysilicon wires can be placed nearby to improve etch uniformity.

2.4.9 Summary

Although the physics of very small transistors is complicated, the impact of nonideal I-V behavior is fairly easy to understand from the designer's viewpoint.

Threshold drops Pass transistors suffer a threshold drop when passing the wrong value: nMOS transistors only pull up to $V_{DD} - V_{tn}$, while pMOS transistors only pull down to $|V_{tp}|$. The magnitude of the threshold drop is increased by the body effect. Therefore, pass transistors do not operate very well in modern processes where the threshold voltage is a significant fraction of the supply voltage. Fully complementary transmission gates should be used where both 0's and 1's must be passed well.

Leakage current Ideally, complementary CMOS gates draw zero current and dissipate zero power when idle. Real gates draw some leakage current. The most important source at this time is subthreshold leakage between source and drain of a transistor that should be cut off. The subthreshold current of an OFF transistor decreases by an order of magnitude for every 60–100 mV that V_{gs} is below V_t. Threshold voltages have been decreasing, so subthreshold leakage has been increasing dramatically. Some processes offer multiple choices of V_t: low-V_t devices are used for high performance in critical circuits, while high-V_t devices are used for low leakage elsewhere.

The transistor gate is a good insulator. However, tunneling current flows through very thin gates. The significance of tunneling is also increasing exponentially and is becoming another important source of leakage current.

Leakage current causes CMOS gates to consume power when idle. It also limits the amount of time that data is retained in dynamic logic, latches, and memory cells. In modern processes, dynamic logic and latches require some sort of feedback to prevent data loss from leakage. Leakage increases at high temperature.

V_{DD} Velocity saturation and mobility degradation result in less current than expected at high voltage. This means that there is no point in trying to use a high V_{DD} to achieve fast transistors, so V_{DD} has been decreasing with process generation to reduce power consumption. Moreover, the very short channels and thin gate oxides would be damaged by high V_{DD}.

Delay Transistors in series drop part of the voltage across each transistor and thus experience smaller fields and less velocity saturation than single transistors. Therefore, series transistors tend to be a bit faster than a simple model would predict. For example, two nMOS transistors in series deliver more than half the current of a single nMOS transistor of the same width. This effect is more pronounced for nMOS transistors than pMOS transistors because nMOS transistors have higher mobility to begin with and thus are more velocity saturated.

Matching If two transistors should behave identically, both should have the same dimensions and orientation and be interdigitated if possible.

2.5 DC Transfer Characteristics

Digital circuits are merely analog circuits used over a special portion of their range. The DC transfer characteristics of a circuit relate the output voltage to the input voltage, assuming the input changes slowly enough that capacitances have plenty of time to charge or discharge. Specific ranges of input and output voltages are defined as valid '0' and '1' logic levels. This section explores the DC transfer characteristics of CMOS gates and pass transistors.

2.5.1 Complementary CMOS Inverter DC Characteristics

Let us derive the DC transfer function (V_{out} vs. V_{in}) for the complementary CMOS inverter shown in Figure 2.23. We begin with Table 2.2, which outlines various regions of operation for the n- and p-transistors. In this table, V_{tn} is the threshold voltage of the n-channel device, and V_{tp} is the threshold voltage of the p-channel device. Note that V_{tp} is negative. The equations are given both in terms of V_{gs}/V_{ds} and V_{in}/V_{out}. As the source of the nMOS transistor is grounded, $V_{gsn} = V_{in}$ and $V_{dsn} = V_{out}$. As the source of the pMOS transistor is tied to V_{DD}, $V_{gsp} = V_{in} - V_{DD}$ and $V_{dsp} = V_{out} - V_{DD}$.

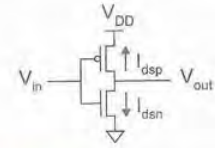


FIG 2.23 A CMOS inverter

Table 2.2 Relationships between voltages for the three regions of operation of a CMOS inverter

	Cutoff	Linear	Saturated
nMOS	$V_{gsn} < V_{tn}$	$V_{gsn} > V_{tn}$	$V_{gsn} > V_{tn}$
	$V_{in} < V_{tn}$	$V_{in} > V_{tn}$	$V_{in} > V_{tn}$
		$V_{dsn} < V_{gsn} - V_{tn}$	$V_{dsn} > V_{gsn} - V_{tn}$
		$V_{out} < V_{in} - V_{tn}$	$V_{out} > V_{in} - V_{tn}$
pMOS	$V_{gsp} > V_{tp}$	$V_{gsp} < V_{tp}$	$V_{gsp} < V_{tp}$
	$V_{in} > V_{tp} + V_{DD}$	$V_{in} < V_{tp} + V_{DD}$	$V_{in} < V_{tp} + V_{DD}$
		$V_{dsp} > V_{gsp} - V_{tp}$	$V_{dsp} < V_{gsp} - V_{tp}$
		$V_{out} > V_{in} - V_{tp}$	$V_{out} < V_{in} - V_{tp}$

The objective is to find the variation in output voltage (V_{out}) as a function of the input voltage (V_{in}). This may be done graphically, analytically (see Exercise 2.16), or through simulation [Carr72]. Given V_{in}, we must find V_{out} subject to the constraint that $I_{dsn} = |I_{dsp}|$. For simplicity, we assume $V_{tp} = -V_{tn}$ and that the pMOS transistor is 2–3 times as wide as the nMOS transistor so $\beta_n = \beta_p$. We relax this assumption in Section 2.5.2.

We commence with the graphical representation of the simple algebraic equations described by EQs (2.5) and (2.8) for the two inverter transistors shown in Figure 2.24(a). The plot shows I_{dsn} and I_{dsp} in terms of V_{dsn} and V_{dsp} for various values of V_{gsn} and V_{gsp}. Figure 2.24(b) shows the same plot of I_{dsn} and $|I_{dsp}|$ now in terms of V_{out} for various values of V_{in}. The possible operating points of the inverter, marked with dots, are the values of V_{out} where $I_{dsn} = |I_{dsp}|$ for a given value of V_{in}. These operating points are plotted on V_{out} vs. V_{in} axes in Figure 2.24(c) to show the inverter DC transfer characteristics. The supply current $I_{DD} = I_{dsn} = |I_{dsp}|$ is also plotted against V_{in} in Figure 2.24(d) showing that both transistors are momentarily ON as V_{in} passes through voltages between GND and V_{DD}, resulting in a pulse of current drawn from the power supply.

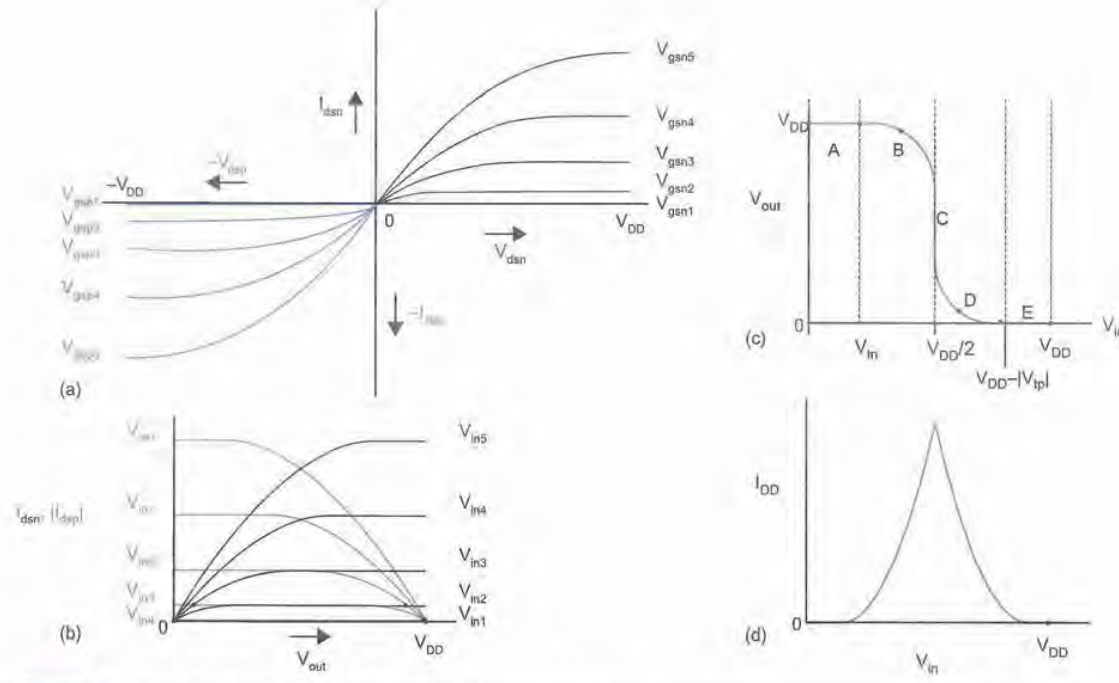


FIG 2.24 Graphical derivation of CMOS inverter DC characteristic

The operation of the CMOS inverter can be divided into five regions indicated on Figure 2.24(c). The state of each transistor in each region is shown in Table 2.3. In region *A*, the nMOS transistor is OFF so the pMOS transistor pulls the output to V_{DD}. In region *B*, the nMOS transistor starts to turn ON, pulling the output down. In region *C*, both transistors are in saturation. Notice that ideal transistors are only in region *C* for $V_{in} = V_{DD}/2$ and that the slope of the transfer curve in this example is −8 in this region, corresponding to infinite gain. Real transistors have finite output resistances on account of channel length modulation, described in Section 2.4.2, and thus have finite slopes over a broader region *C*. In region *D*, the pMOS transistor is partially ON and in region *E*, it is completely OFF, leaving the nMOS transistor to pull the output down to GND. Also notice that the inverter's current consumption is zero when the input is within a threshold voltage of the V_{DD} or GND rails. This feature is important for low-power operation.

Table 2.3 Summary of CMOS inverter operation

Region	Condition	p-device	n-device	Output
A	$0 \le V_{in} < V_{tn}$	linear	cutoff	$V_{out} = V_{DD}$
B	$V_{tn} \le V_{in} < V_{DD}/2$	linear	saturated	$V_{out} > V_{DD}/2$
C	$V_{in} = V_{DD}/2$	saturated	saturated	V_{out} drops sharply
D	$V_{DD}/2 < V_{in} \le V_{DD} - \lvert V_{tp}\rvert$	saturated	linear	$V_{out} < V_{DD}/2$
E	$V_{in} > V_{DD} - \lvert V_{tp}\rvert$	cutoff	linear	$V_{out} = 0$

Figure 2.25 shows simulation results of an inverter from a 180 nm process. The pMOS transistor is twice as wide as the nMOS transistor to achieve approximately equal betas. Simulation matches the simple models reasonably well, although the transition is not quite as steep on account of channel length modulation.

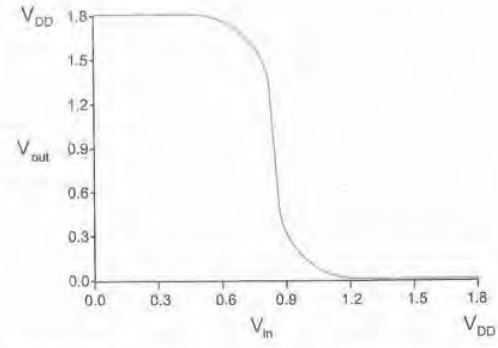


FIG 2.25 Simulated CMOS inverter DC characteristic

The crossover point where $V_{inv} = V_{in} = V_{out}$ is called the *input threshold*. Because both mobility and the magnitude of the threshold voltage decrease with temperature for nMOS and pMOS transistors, the input threshold of the gate is only weakly sensitive to temperature.

2.5.2 Beta Ratio Effects

We have seen that for $\beta_p = \beta_n$, the inverter threshold voltage V_{inv} is $V_{DD}/2$. This may be desirable because it maximizes noise margins (see Section 2.5.3) and allows a capacitive load to charge and discharge in equal times by providing equal current source and sink capabilities (see Section 4.2). Inverters with different beta ratios β_p/β_n are called *skewed* inverters [Sutherland99]. If $\beta_p/\beta_n > 1$ (e.g., 2), the inverter is *HI-skewed*. If $\beta_p/\beta_n < 1$ (e.g., 1/2) the inverter is *LO-skewed*. If $\beta_p/\beta_n = 1$, the inverter has normal skew or is *unskewed*.

A HI-skew inverter has a stronger pMOS transistor. Therefore, if the input is $V_{DD}/2$, we would expect the output will be greater than $V_{DD}/2$. In other words, the input threshold must be higher than for an unskewed inverter. Similarly a LO-skew inverter has a weaker pMOS transistor and thus a lower switching threshold.

Figure 2.26 explores the impact of skewing the beta ratio on the DC transfer characteristics. As the beta ratio is changed, the switching threshold moves. However, the output voltage transition remains sharp. Gates are usually skewed by adjusting the widths of transistors while maintaining minimum length for speed.

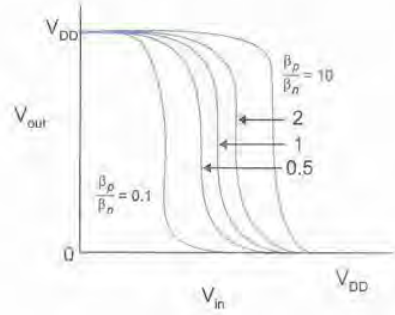


FIG 2.26 Transfer characteristics of skewed inverters

DC transfer characteristics of other complementary CMOS gates can be understood by collapsing the gates into an equivalent inverter. Series transistors can be viewed as a single transistor of greater length. If only one of several parallel transistors is ON, the other transistors can be ignored. If several parallel transistors are ON, the collection can be viewed as a single transistor of greater width.

2.5.3 Noise Margin

Noise margin is closely related to the DC voltage characteristics [Wakerly00]. This parameter allows you to determine the allowable noise voltage on the input of a gate so that the output will not be corrupted. The specification most commonly used to describe noise margin (or *noise immunity*) uses two parameters: the *LOW* noise margin, NM_L, and the *HIGH* noise margin, NM_H. With reference to Figure 2.27, NM_L is defined as the difference in maximum LOW input voltage recognized by the receiving gate and the maximum LOW output voltage produced by the driving gate.

$$NM_L = V_{IL} - V_{OL} \tag{2.42}$$

The value of NM_H is the difference between the minimum HIGH output voltage of the driving gate and the minimum HIGH input voltage recognized by the receiving gate. Thus

$$NM_H = V_{OH} - V_{IH} \tag{2.43}$$

where

V_{IH} = minimum HIGH input voltage
V_{IL} = maximum LOW input voltage
V_{OH} = minimum HIGH output voltage
V_{OL} = maximum LOW output voltage.

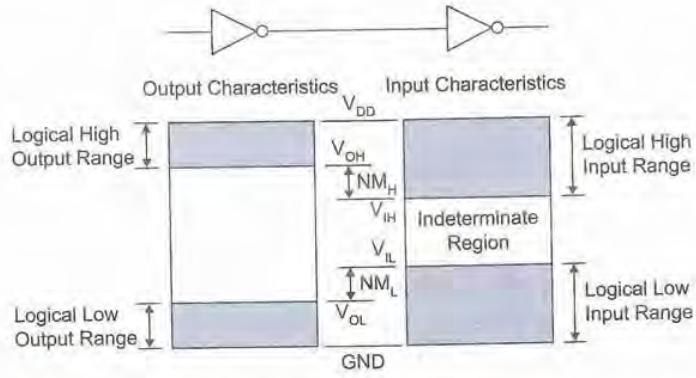


FIG 2.27 Noise margin definitions

Inputs between V_{IL} and V_{IH} are said to be in the *indeterminate region* or *forbidden zone* and do not represent legal digital logic levels. Therefore, it is generally desirable to have V_{IH} as close as possible to V_{IL} and for this value to be midway in the "logic swing," V_{OL} to

V_{OH}. This implies that the transfer characteristic should switch abruptly, that is, there should be high gain in the transition region. For the purpose of calculating noise margins, the transfer characteristic of the inverter and the definition of voltage levels V_{IL}, V_{OL}, V_{IH}, V_{OH} are shown in Figure 2.28. Logic levels are defined at the unity gain point where the slope is −1. This gives a conservative bound on the worst-case static noise margin [Hill68, Lohstroh83, Shepard99]. For the inverter shown, the NM_L is 0.46 V_{DD} while the NM_H is 0.34 V_{DD}. Note that the output is slightly degraded when the input is at its worst legal value; this is called *noise feedthrough* or *propagated noise*. The exercises at the end of the chapter examine graphical and analytical approaches of finding the logic levels and noise margins.

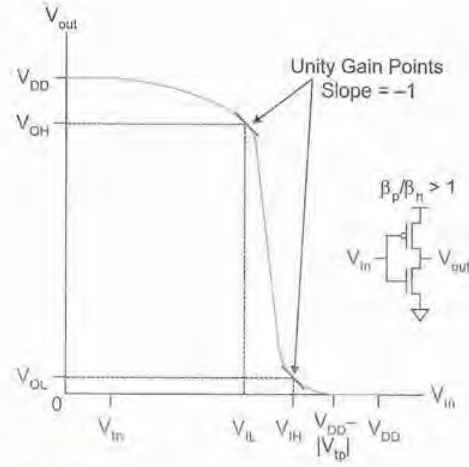


FIG 2.28 CMOS inverter noise margins

Note that if $|V_{tp}| = V_{tn}$, then NM_H and NM_L increase as threshold voltages are increased. If either NM_L or NM_H for a gate are too small (e.g., below about 0.1 V_{DD}), then the gate may be disturbed by noise that occurs on the inputs. Quite often, noise margins are compromised to improve speed. Circuit examples in Chapter 6 will illustrate this tradeoff. Noise sources tend to scale with the supply voltage, so noise margins are best given as a fraction of the supply voltage. A noise margin of 0.4 V is quite comfortable in a 1.8 V process, but marginal in a 5 V process.

DC analysis gives us the *static noise margins* specifying the level of noise that a gate may see for an indefinite duration. Larger noise pulses may be acceptable if they are brief; these are described by *dynamic noise margins* specified by a maximum amplitude as a function of the duration [Lohstroh79, Somasekhar00]. Unfortunately, there is no simple amplitude-duration product that conveniently specifies dynamic noise margins.

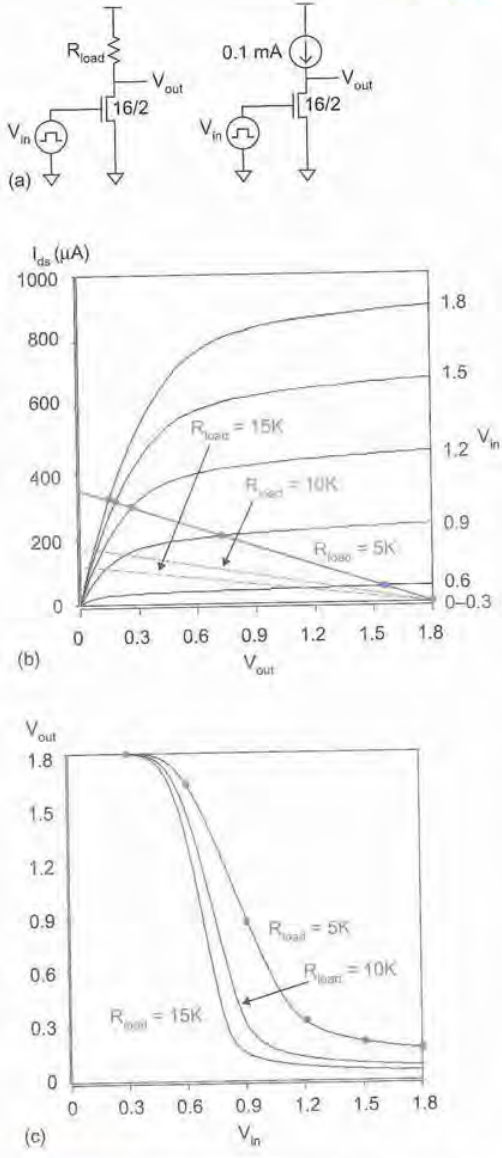


FIG 2.29 Generic nMOS inverters with resistive or constant current load

OPTIONAL

2.5.4 Ratioed Inverter Transfer Function

Apart from the complementary CMOS inverter, there are other forms of MOS inverter that can be used to build logic gates. Figure 2.29(a) shows a generic nMOS inverter that uses either a resistive load or a constant current source. For the resistor case, if we superimpose the resistor load line on the I-V characteristics of the pulldown transistor (Figure 2.29(b)), we can see that at $V_{in} = V_{DD}$, the output is some small V_{out} (V_{OL}) (Figure 2.29(c)). When $V_{in} = 0$, V_{out} rises to V_{DD}. As the resistor is made larger, the V_{OL} decreases and the current flowing when the inverter is turned on decreases. Correspondingly, as the load resistor is decreased in value, the V_{OL} rises and the ON current rises. Selection of the resistor value would seek a compromise between V_{OL}, the current drawn, and the pullup delay that increases with the value of the load resistor. Current sources have high output resistance and thus offer sharper transitions.

Neither high-value resistors nor ideal current sources are readily available in most CMOS processes. A more practical circuit called a *pseudo-nMOS inverter* is shown in Figure 2.30(a). It uses a pMOS transistor pullup or *load* that has its gate permanently grounded to approximate a constant current source. Pseudo-nMOS circuits get their name from the early nMOS technology (which preceded CMOS technology as a major systems technology) in which only nMOS transistors were available; the grounded pMOS transistor is reminiscent of a depletion mode nMOS transistor that is always ON.

The transfer characteristics may again be derived by finding V_{out} for which $I_{dn} = |I_{dp}|$ for a given V_{in}, as shown in Figure 2.30(b) and Figure 2.30(c). The beta ratio affects the shape of the transfer characteristics and the V_{OL} of the inverter. Larger pMOS transistors offer faster rise times but less sharp transfer characteristics. Figure 2.30(d) shows that when the nMOS transistor is turned on, a constant DC current flows in the circuit.

The gates in this section are called *ratioed* circuits because the transfer function depends on the ratio of the strength of the pulldown transistor to the pullup device. The resistor, current source, or ON transistor is sometimes called a *static load*. It is possible to construct other ratioed circuits such as NAND or NOR gates by replacing the pullup transistors with a single pullup device. Unlike complementary circuits, the ratio must be chosen so the circuit operates correctly despite any variations from nominal component values that may occur during manufacturing. Moreover, ratioed circuits dissipate power continually in certain states (e.g., when the output is low) and have poorer noise margins than complementary circuits.

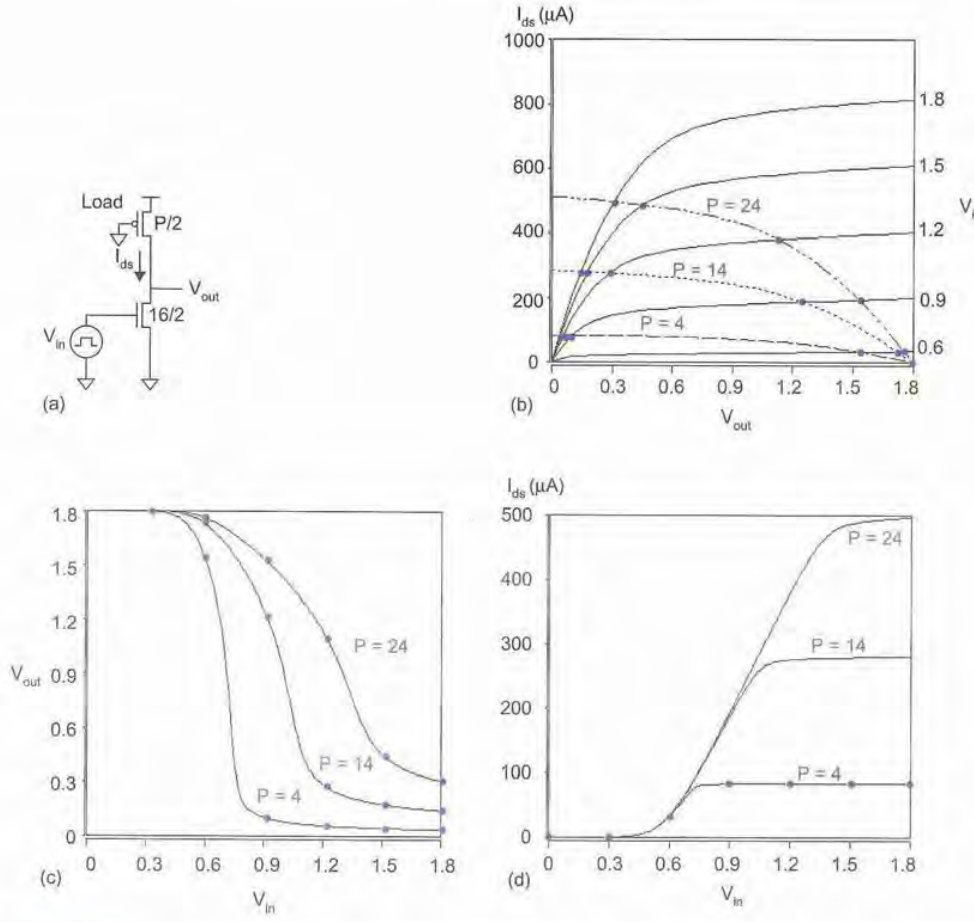


FIG 2.30 Pseudo-nMOS inverter and DC transfer characteristics

Therefore, ratioed circuits tend to be used only in very limited circumstances where they offer critical benefits such as smaller area or reduced input capacitance. We will return to ratioed circuits in Section 6.2.2.

2.5.5 Pass Transistor DC Characteristics

Recall from Section 1.4.6 that nMOS transistors pass '0's well but '1's poorly. We are now ready to better define "poorly." Figure 2.31(a) shows an nMOS transistor with the gate and drain tied to V_{DD}. Imagine that the source is initially at $V_s = 0$. $V_{gs} > V_{tn}$, so the transis-

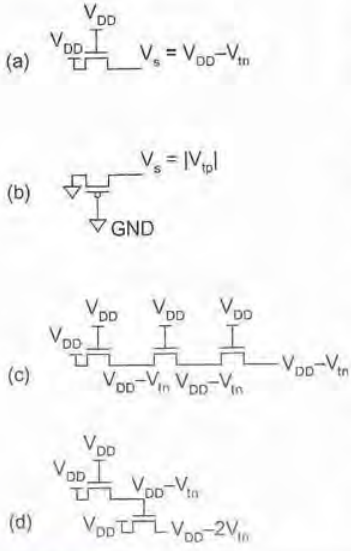


FIG 2.31 Pass transistor threshold drops

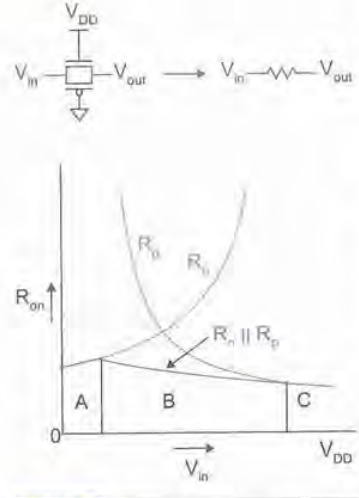


FIG 2.32 Resistance of a transmission gate as a function of input voltage

tor is ON and current flows. If the voltage on the source rises to $V_s = V_{DD} - V_{tn}$, V_{gs} falls to V_{tn} and the transistor cuts itself OFF. Therefore, nMOS transistors attempting to pass a '1' never pull the source above $V_{DD} - V_{tn}$.[7] This loss is sometimes called a *threshold drop*.

Moreover, when the source of the nMOS transistor rises, V_{sb} becomes nonzero. As described in Section 2.4.2, this nonzero source to body potential introduces the body effect that increases the threshold voltage. Using the data from the example in that section, a pass transistor driven with $V_{DD} = 5$ V would produce an output of only 3.34 V, potentially violating the noise margins of the next stage.

Similarly, pMOS transistors pass '1's well but '0's poorly. If the pMOS source drops below $|V_{tp}|$, the transistor cuts off. Hence, pMOS transistors only pull down to within a threshold above GND, as shown in Figure 2.31(b).

As the source can rise to within a threshold voltage of the gate, the output of several transistors in series is no more degraded than that of a single transistor (Figure 2.31(c)). However, if a degraded output drives the gate of another transistor, the second transistor can produce an even further degraded output (Figure 2.31(d)).

Also recall from Section 1.5.6 that a transmission gate consists of an nMOS transistor and a pMOS transistor in parallel with gates controlled by complementary signals. When the transmission gate is ON, at least one of the two transistors is ON for any output voltage and hence the transmission gate passes both '0's and '1's well. The transmission gate is a fundamental and ubiquitous component in MOS logic. It finds use as a multiplexing element, a logic structure, a latch element, and an analog switch. The transmission gate acts as a voltage-controlled resistor connecting the input and the output.

Figure 2.32 plots the transmission gate ON resistance as the input voltage is swept from GND to V_{DD}, assuming the output voltage closely follows. In region *A*, the nMOS transistor is operating linearly and the pMOS is cut off. In region *B*, both transistors are linear. In region *C*, the nMOS transistor is cut off and the pMOS is linear. If both transistors are of equal size, the characteristics are slightly asymmetric because of the better mobility of the nMOS transistor. The effective ON resistance is the parallel combination of the two resistances and is relatively constant across the full range of input voltages.

2.5.6 Tristate Inverter

By cascading a transmission gate with an inverter, the tristate inverter shown in Figure 2.33(a) is constructed. When $EN = 0$ and $ENb = 1$, the output of the inverter is in a tristate condition (the *Y* output is not driven by the *A* input). When $EN = 1$ and $ENb = 0$, the *Y* output is equal to the complement of *A*. The connection between the n- and p-driver transistors can be omitted (Figure

[7] Technically, the output can rise higher very slowly on account of subthreshold current.

2.33(b-c)) and the operation remains substantially the same. For the same size n- and p-devices, this tristate inverter is approximately half the speed of a complementary CMOS inverter. The tristate inverter forms the basis for various types of clocked logic, latches, bus drivers, multiplexers, and I/O structures. The circuit in Figure 2.33(d) interchanges the *A* and enable terminals. It is logically equivalent but electrically inferior because if the output is tristated but *A* toggles, charge from the internal nodes may disturb the floating output node (see Section 6.3.4 for more discussion of charge sharing).

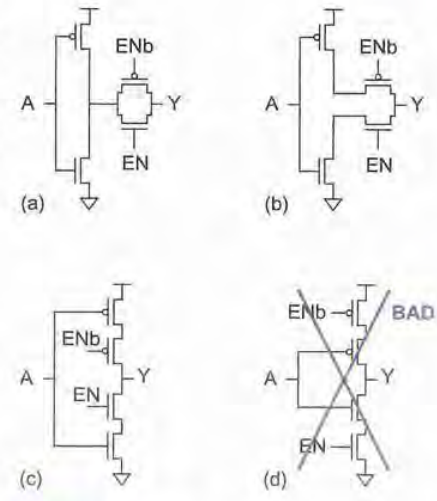


FIG 2.33 Tristate inverter

2.6 Switch-level RC Delay Models

It is very useful to be able to estimate the delay of a gate or compare circuit topologies without resorting to circuit simulation. RC delay models provide a means to make such approximate calculations. They approximate the nonlinear transistor I-V and C-V characteristics with an average resistance and capacitance over the switching range of the gate; this works remarkably well for delay estimation despite its obvious problems in predicting detailed analog behavior.

The RC delay model treats transistors as switches in series with resistors. Define a unit nMOS transistor to have *effective* resistance *R*. The size of the unit transistor is arbitrary but conventionally refers to a transistor with minimum length and minimum contacted diffusion width (i.e., 4/2 λ). An nMOS transistor of *k* times unit width has resistance *R/k*. A unit pMOS transistor has greater resistance, generally in the range of 2*R* – 3*R*, because of its lower mobility. Throughout this book we will use 2*R* for examples to keep arithmetic simple.

The resistance at some operating point can be defined as:

$$R = \left(\frac{\partial I_{ds}}{\partial V_{ds}} \right)^{-1} \tag{2.44}$$

If V_{ds} is small and the transistor is operating in the linear region, the resistance can be approximated by differentiating EQ (2.5) as

$$R = \frac{1}{\beta\left(V_{gs} - V_t\right)} = \frac{1}{\mu C_{ox}} \frac{L}{W} \frac{1}{\left(V_{gs} - V_t\right)} \tag{2.45}$$

This formula is not particularly useful for calculations because the current rolls off (and resistance increases) as V_{ds} increases; we are more interested in some average current over the switching transition. Moreover, velocity saturation and other effects are important in

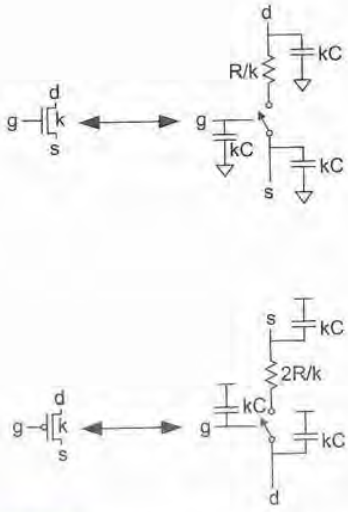


FIG 2.34 Equivalent RC circuit models

modern transistors, so estimating resistance from fundamental principles is not very practical. Nevertheless, EQ (2.45) shows that resistance is proportional to L/W and decreases with V_{gs}. A better way to determine resistance is to simulate a transistor driving a known capacitance and measure the time constant (see Section 5.4.5).

Each transistor also has gate and diffusion capacitance. We define C to be the gate capacitance of a unit transistor of either flavor. A transistor of k times unit width has capacitance kC. Diffusion capacitance depends on the size of the source/drain region. Using the approximations from Section 2.3.1, we assume the source or drain of a unit transistor to also have capacitance C. Wider transistors have proportionally greater diffusion capacitance. As discussed in Section 2.3.1, C is typically in the range of 1.5–2 fF/μm of transistor width for a minimum length transistor (2 fF/μm is a reasonable value for rough estimates).

Figure 2.34 shows equivalent RC circuit models for nMOS and pMOS transistors of width k with contacted diffusion on both source and drain. The pMOS capacitors are shown with V_{DD} as their second terminal because the n-well is usually tied high. However, the behavior of the capacitor from a delay perspective is independent of the second terminal voltage so long as it is constant. Hence, we sometimes draw the second terminal as ground for convenience.

The propagation delay of a logic gate can be estimated from the RC models. Figure 2.35 shows how to estimate the delay of a fanout-of-1 inverter. The unit inverter of Figure 2.35(a) is composed from an nMOS transistor of unit size and a pMOS transistor of twice unit width to achieve equal rise and fall resistance. (Remember: If the nMOS mobility were three times that of the pMOS transistor, the pMOS device would have to be made three times wider than the nMOS transistor to achieve symmetric rise and fall times.) Note that if you use a cell library with a larger minimum-sized inverter, it is handy to define the unit transistor as the nMOS transistor in the basic inverter. Figure 2.35(b) gives an equivalent circuit, showing the first inverter driving the second inverter's gate. If the input A rises, the nMOS transistor will be ON and the pMOS OFF. Figure 2.35(c) illustrates this case with the switches removed. The capacitors shorted between two constant supplies are also removed because they are not charged or discharged. The propagation delay of $t_{pd} = R \cdot (6C) = 6RC$ is estimated as the RC time constant of the resistor discharging the diffusion and load capacitances. If an ideal inverter could be constructed with no parasitic diffusion capacitance, the delay would be only $3RC$. This delay of an ideal inverter with no parasitics driving an identical inverter is a figure of merit describing a manufacturing process and is sometimes called τ [Sutherland99].[8] Delay estimates for more complex gates are examined in Section 4.2.

[8] Do not confuse this definition of $\tau = 3RC$ with Mead and Conway's definition [Mead80] $\tau = RC$, the delay of an nMOS transistor driving its own gate.

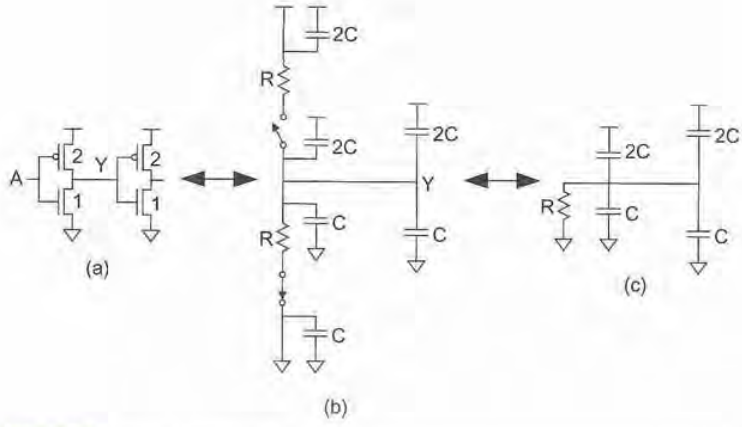


FIG 2.35 Inverter propagation delay

The effective resistance of a transistor passing a value in its poor direction is greater. For example, we can model the effective resistance of a unit nMOS transistor passing a '1' as $2R$ and a unit pMOS transistor passing a '0' as $4R$. The effective resistance of a transmission gate is the parallel combination of the resistances of the two transistors. Figure 2.36 shows that the effective resistance passing a '0' is $R \,||\, 4R = (4/5)R$. The effective resistance passing a '1' is $2R \,||\, 2R = R$. Hence, a transmission gate made from unit transistors is approximately R in either direction. Note that transmission gates are commonly built using equal-sized nMOS and pMOS transistors. Boosting the size of the pMOS transistor only slightly improves the effective resistance while significantly increasing the capacitance.

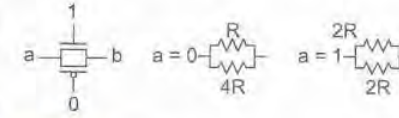


FIG 2.36 Effective resistance of a unit transmission gate

Most digital circuits use minimum-length transistors. If the length is greater than minimum, both resistance and capacitance to first order grow linearly with channel length. Section 5.4 describes how to construct SPICE simulations to extract R and C for a particular fabrication process operating at a particular power supply.

Velocity saturation also impacts the effective resistance. If velocity saturation were not a factor, the effective resistance of two transistors in series would be double that of a single transistor of equal size. If velocity saturation is significant, series transistors each experience a smaller V_{ds} and hence are less velocity saturated. Therefore, the resistance of series transistors is often slightly less than our simple model predicts. The effect is more pronounced for nMOS transistors than pMOS because of the higher mobility and greater degree of velocity saturation.

2.7 Pitfalls and Fallacies

This section lists a number of pitfalls and fallacies that can deceive the novice (or experienced) designer.

Blindly trusting one's models

Models should be viewed as only approximations to reality, not reality itself, and used within their limitations. In particular, simple models like the Shockley or RC models aren't even close to accurate fits for the I-V characteristics of a modern transistor. They are valuable for the insight they give on trends (i.e., making a transistor wider increases its gate capacitance and decreases its ON resistance), not for the absolute values they predict. Cutting-edge projects often target processes that are still under development, so these models should only be viewed as speculative. Finally, processes may not be fully characterized over all operating regimes; for example, don't assume that your models are accurate in the subthreshold region unless your vendor tells you so. Having said this, modern SPICE models do an extremely good job of predicting performance well into the GHz range for well characterized processes and models when using proper design practices (such as accounting for temperature, voltage, and process variation).

Using excessively complicated models for manual calculations

Because models cannot be perfectly accurate, there is little value in using excessively complicated models, particularly for hand calculations. Simpler models give more insight on key tradeoffs and more rapid feedback during design. Moreover, RC models calibrated against simulated data for a fabrication process can estimate delay just as accurately as elaborate models based on a large number of physical parameters but not calibrated to the process.

Assuming a transistor with twice the drawn length has exactly half the current

To first order, current is proportional to W/L. In modern transistors, the effective transistor length is usually shorter than the drawn length, so doubling the drawn length reduces current by more than a factor of two. Moreover, the threshold voltage tends to increase for longer transistors, resulting in less current. Therefore, it is a poor strategy to try to ratio currents by ratioing transistor lengths.

Assuming two transistors in series deliver exactly half the current of a single transistor

To first order, this would be true. However, two transistors in series each see a smaller electric field across the channel and hence are each less velocity saturated. Therefore, two series transistors in a modern process will deliver more than half the current of a single transistor. This is more pronounced for nMOS than pMOS transistors because of the higher mobility and the higher degree of velocity saturation of electrons than holes at a given field. This means that NAND gates perform better than first order estimates might predict.

Using nMOS pass transistors

nMOS pass transistors only pull up to $V_{DD} - V_t$. This voltage may fall below V_{IH} of a receiver, especially as V_{DD} decreases. For example, one author worked with a scan latch containing an nMOS pass transistor that operated correctly in a 250 nm process at 2.5 V. When the latch was ported to a 180 nm process at 1.8 V, the scan chain stopped working. The problem was traced to the pass transistor and the scan chain was made operational in the lab by raising V_{DD} to 2 V. A better solution is to use transmission gates in place of pass transistors.

Summary

In summary, we have seen that MOS transistors are four-terminal devices with a gate, source, drain, and body. In normal operation, the body is tied to GND or V_{DD} so the transistor can be modeled as a three-terminal device. The transistor behaves as a voltage-controlled switch. An nMOS switch is OFF (no path from source to drain) when the gate voltage is below some threshold V_t. The switch turns ON, forming a channel connecting source to drain, when the gate voltage rises above V_t. This chapter has developed more elaborate models to predict the amount of current that flows when the transistor is ON. The transistor operates in three modes depending on the terminal voltage:

- $V_{gs} < V_t$ Cutoff $I_{ds} \sim 0$
- $V_{gs} > V_t$, $V_{ds} < V_{gs} - V_t$ Linear I_{ds} increases with V_{ds} (like a resistor)
- $V_{gs} > V_t$, $V_{ds} > V_{gs} - V_t$ Saturation I_{ds} constant (like a current source)

In an ideal transistor, the saturation current depends on $(V_{gs} - V_t)^2$. pMOS transistors are similar to nMOS transistors, but have the signs reversed and deliver about half the current because of lower mobility. From the ideal model, we can derive the DC transfer characteristics and noise margins of logic gates using either the analytical expressions or a graphical load line analysis or simulation.

In a real transistor, the I-V characteristics are more complicated. Modern transistors are extraordinarily small and thus experience enormous electric fields even at low voltage. The high fields cause velocity saturation and mobility degradation that lead to less current than you might otherwise expect. This can be modeled as a saturation current dependent on $(V_{gs} - V_t)^{\alpha}$, where the velocity saturation index α is less than 2. Moreover, the saturation current does increase slightly with V_{ds} because of channel length modulation. Although simple hand calculations are no longer accurate, the general shape does not change very much and the transfer characteristics can still be derived using the graphical or simulation methods.

Even when the gate voltage is low, the transistor is not completely OFF. Subthreshold current through the channel drops off exponentially for $V_{gs} < V_t$, but is nonnegligible for transistors with low thresholds. Junction leakage currents flow through the reverse-biased p–n junctions. Tunneling current flows through the insulating gate when the oxide becomes thin enough.

Unlike ideal switches, MOS transistors pass some voltage levels better than others. An nMOS transistor passes '0's well, but only pulls up to $V_{DD} - V_{tn}$ when passing '1's. The pMOS passes '1's well, but only pulls down to $|V_{tp}|$ when passing '0's. This threshold drop is exacerbated by the body effect, which increases the threshold voltage when the source is at a different potential than the body.

Transistor speed depends on the ratio of current to capacitance. The two main sources of capacitance in a transistor are the gate capacitance formed by the thin gate oxide and

the diffusion capacitance formed by the depletion regions between the source or drain and body. The diffusion capacitance is voltage-dependent, but can be modeled across the digital switching voltages with an average value.

Circuit simulators use very elaborate models to calculate the currents and voltages in a transistor. These models are too complicated to give much insight into the circuit behavior. For the purpose of estimating delay, we can approximate an ON transistor with an effective resistance such that the product of this resistance and the load capacitance matches the gate delay. This approximation is remarkably good and will be developed further in subsequent chapters to explain why some circuits are faster than others and to help optimize gate delay.

[Muller03] offers a comprehensive treatment of device physics at a more advanced level. [Gray01] describes MOSFET models in more detail from the analog designer's point of view.

Exercises

2.1 Consider an nMOS transistor in a 0.6 μm process with $W/L = 4/2\ \lambda$ (i.e., 1.2/0.6 μm). In this process, the gate oxide thickness is 100 Å and the mobility of electrons is 350 $cm^2/V \cdot s$. The threshold voltage is 0.7 V. Plot I_{ds} vs. V_{ds} for $V_{gs} = 0, 1, 2, 3, 4,$ and 5 V.

2.2 Show that the current through two transistors in series is equal to the current through a single transistor of twice the length if the transistors are well described by the Shockley model. Specifically show that $I_{DS1} = I_{DS2}$ in Figure 2.37 when the transistors are in their linear region: $V_{DS} < V_{DD} - V_t$, $V_{DD} > V_t$ (this is also true in saturation). *Hint:* Express the currents of the series transistors in terms of V_1 and solve for V_1.

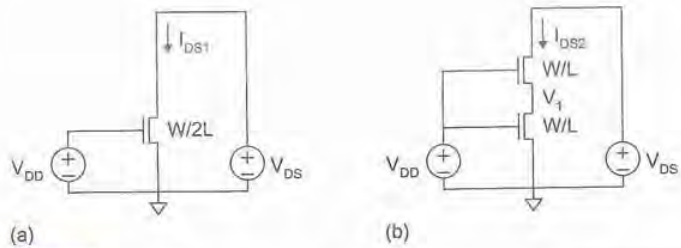


FIG 2.37 Current in series transistors

2.3 In Exercise 2.2, the body effect was ignored. If the body effect is considered, will I_{DS2} be equal to, greater than, or less than I_{DS1}? Explain.

2.4 A 90 nm long transistor has a gate oxide thickness of 16 Å. What is its gate capacitance per micron of width?

2.5 Calculate the diffusion parasitic C_{db} of the drain of a unit-sized contacted nMOS transistor in a 0.6 μm process when the drain is at 0 and at V_{DD} = 5 V. Assume the substrate is grounded. The transistor characteristics are CJ = 0.42 fF/μm², MJ = 0.44, $CJSW$ = 0.33 fF/μm, $MJSW$ = 0.12, and ψ_0 = 0.98 V at room temperature.

2.6 Consider the nMOS transistor in a 0.6 μm process with gate oxide thickness of 100 Å. The doping level is $N_A = 2 \cdot 10^{17}$ cm^{-3} and the nominal threshold voltage is 0.7 V. The body is tied to ground with a substrate contact. How much does the threshold change at room temperature if the source is at 4 V instead of 0?

2.7 Does the body effect of a process limit the number of transistors that can be placed in series in a CMOS gate at low frequencies?

2.8 Sometimes the substrate is connected to a voltage called the substrate bias to alter the threshold of the nMOS transistors. If the threshold of an nMOS transistor is to be raised, should a positive or negative substrate bias be used?

2.9 An nMOS transistor has a threshold voltage of 0.4 V and a supply voltage of V_{DD} = 1.2 V. A circuit designer is evaluating a proposal to reduce V_t by 100 mV to obtain faster transistors.

a) By what factor would the saturation current increase (at $V_{gs} = V_{ds} = V_{DD}$) if the transistor were ideal?

b) By what factor would the subthreshold leakage current increase at room temperature at V_{gs} = 0? Assume n = 1.4.

c) By what factor would the subthreshold leakage current increase at 120° C? Assume the threshold voltage is independent of temperature.

2.10 As temperature rises, does the current through an ON transistor increase or decrease? Does current through an OFF transistor increase or decrease? Will a chip operate faster at high temperature or low temperature? Explain.

2.11 Find the subthreshold leakage current of an inverter at room temperature if the input A = 0. Let $\beta_n = 2\beta_p$ = 1 mA/V², n = 1.4, and $|V_t|$ = 0.4 V. Assume the body effect and DIBL coefficients are $\gamma = \eta = 0$.

2.12 Repeat Exercise 2.11 for a NAND gate built from unit transistors with inputs $A = B$ = 0. Show that the subthreshold leakage current through the series transistors is half that of the inverter.

2.13 Repeat Exercises 2.11 and 2.12 when η = 0.04, as in the case of a more realistic transistor. γ has a secondary effect, so assume that it is 0. Did the leakage currents go up or down in each case? Is the leakage through the series transistors more than half, exactly half, or less than half of that through the inverter?

2.14 Peter Pitfall is offering to license to you his patented noninverting buffer circuit shown in Figure 2.38. Graphically derive the transfer characteristics for this buffer. Why is it a bad circuit idea?

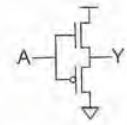


FIG 2.38 Noninverting buffer

2.15 A novel inverter has the transfer characteristics shown in Figure 2.39. What are the values of V_{IL}, V_{IH}, V_{OL}, and V_{OH} that give best noise margins? What are these high and low noise margins?

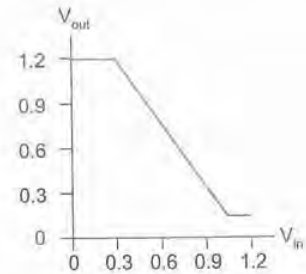


FIG 2.39 Transfer characteristics

2.16 Section 2.5.1 graphically determined the transfer characteristics of a static CMOS inverter. Derive analytic expressions for V_{out} as a function of V_{in} for regions B and D of the transfer function. Let $|V_{tp}| = V_{tn}$ and $\beta_p = \beta_n$.

2.17 Using the results from Exercise 2.16, calculate the noise margin for a CMOS inverter operating at 1.0 V with $V_{tn} = |V_{tp}| = 0.35$ V, $\beta_p = \beta_n$.

2.18 Repeat Exercise 2.16 if the thresholds and betas of the two transistors are not necessarily equal. Also solve for the value of V_{in} for region C where both transistors are saturated.

2.19 Using the results from Exercise 2.18, calculate the noise margin for a CMOS inverter operating at 1.0 V with $V_{tn} = |V_{tp}| = 0.35$ V, $\beta_p = 0.5\beta_n$.

2.20 Derive V_{out} using the Shockley models for the pseudo-nMOS inverter from Figure 2.30 with $V_{in} = V_{DD}$ as a function of the threshold voltages and beta values of the two transistors. Assume $V_{out} < |V_{tp}|$.

2.21 Give an expression for the output voltage for the pass transistor networks shown in Figure 2.40. Neglect the body effect.

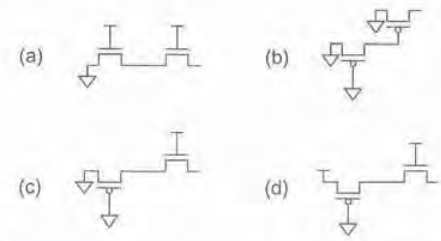


FIG 2.40 Pass transistor networks

2.22 Suppose $V_{DD} = 1.2$ V and $V_t = 0.4$ V. Determine V_{out} in Figure 2.41 for: (a) $V_{in} = 0$ V; (b) $V_{in} = 0.6$ V; (c) $V_{in} = 0.9$ V; and (d) $V_{in} = 1.2$ V. Neglect the body effect.

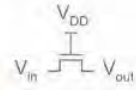


FIG 2.41 Single pass transistor

CMOS Processing Technology

3

3.1 Introduction

Chapter 1 summarized the steps in a basic CMOS process. These steps are expanded upon in this chapter. Where possible, the processing details are related to the way CMOS circuits and systems are designed. In addition, this chapter investigates some emerging technologies that may become important in the future. Modern CMOS processing is complex, and while coverage of every nuance of modern CMOS processing is not within the scope of this book, we focus on the fundamental concepts.

A fair question from a designer would be "Why do I care how transistors are made?" In many cases, if designers understand the physical process, they will comprehend the reason for the underlying design rules and in turn use this knowledge to create a better design. Understanding the manufacturing steps is also important when debugging some difficult chip failures and improving yield.

3.2 CMOS Technologies

3.2.1 Background

In this section we provide an overview of current CMOS technologies with a simplified treatment of the process steps.

The main CMOS technologies are:

- n-well process
- p-well process
- twin-well process
- triple-well process

By adding bipolar transistors (either silicon or silicon germanium—SiGe), the range of processes can be expanded. Silicon-on-insulator processes are also available through some manufacturers (see Section 3.4.1.2).

Chapter 1 outlined an n-well process. Historically, p-well processes preceded n-well processes. In a p-well process, the nMOS transistors are built in a p-well and the pMOS transistor is placed in the n-type substrate. p-well processes were used to optimize the pMOS transistor performance. Improved techniques allowed good pMOS transistors to be fabricated in an n-well and excellent nMOS transistors to be fabricated in the p-type substrate of an n-well process. Twin-well processes accompanied the emergence of n-well processes. A twin-well process allows the optimization of each transistor type. A third well can be added to create a triple-well process. The triple-well process has emerged to provide good isolation between analog and digital blocks in mixed-signal chips; it is also used to isolate high-density dynamic memory from logic. Most fabrication lines provide a baseline twin-well process that can be upgraded to a triple-well process with the addition of a single mask level. In this section we will present a CMOS process in the 130 or 90 nm generation that is representative of current commercial processes.

To review the material covered in Chapter 1, silicon in its pure or *intrinsic* state is a semiconductor, having bulk electrical resistance somewhere between that of a conductor and an insulator. The conductivity of silicon can be raised by several orders of magnitude by introducing *impurity* atoms into the silicon crystal lattice. These dopants can supply either free electrons or holes. Group III impurity elements such as boron that use up electrons are referred to as *acceptors* because they accept some of the electrons already in the silicon, leaving holes. Similarly, Group V *donor* elements such as arsenic and phosphorous provide electrons. Silicon that contains a majority of donors is known as *n-type*, while silicon that contains a majority of acceptors is known as *p-type*. When n-type and p-type materials are brought together, the region where the silicon changes from n-type to p-type is called a *junction* (junctions also can be formed between two areas with different doping). By arranging junctions in certain physical structures and combining them with other physical structures, various semiconductor devices can be constructed. Over the years, silicon semiconductor processing has evolved sophisticated techniques for building these junctions and other insulating and conducting structures.

3.2.2 Wafer Formation

The basic raw material used in modern semiconductor *fabs* (fabrication facilities) is a *wafer* or disk of silicon, which currently varies from roughly 75 mm to 300 mm (12"—a dinner plate!) in diameter and less than 1 mm thick. Wafers are cut from ingots of single-crystal silicon that have been pulled from a crucible melt of pure molten silicon. This is known as the *Czochralski* method and is currently the most common method for producing single-crystal material. Controlled amounts of impurities are added to the melt to provide the crystal with the required electrical properties. A seed crystal is dipped into the melt to initiate crystal growth. The silicon ingot takes on the same crystal orientation as the seed. A graphite radiator heated by radio-frequency induction surrounds the quartz crucible holding the melt and maintains the temperature a few degrees above the melting point of silicon (1425° C). The atmosphere is typically helium or argon to prevent the silicon from oxidizing.

The seed is gradually withdrawn vertically from the melt while simultaneously being rotated. The molten silicon attaches itself to the seed and recrystallizes as it is withdrawn. The seed withdrawal and rotation rates determine the diameter of the ingot. Growth rates vary from 30 to 180 mm/hour.

3.2.3 Photolithography

Recall that regions of dopants, polysilicon, metal, and contacts are defined using masks. For instance, in places covered by the mask, ion implantation might not occur or the dielectric or metal layer might be left intact. In areas where the mask is absent, the implantation can occur, or dielectric or metal could be etched away. The patterning is achieved by a process called *photolithography*, from the Greek *photo* (light), *lithos* (stone), and *graphe* (picture), which literally means "carving pictures in stone using light." The primary method for defining areas of interest (i.e., where we want material to be present or absent) on a wafer is by the use of *photoresists*. The wafer is coated with the photoresist and subjected to selective illumination through the *photomask*. After the initial patterning of photoresist, other barrier layers such as polycrystalline silicon, silicon dioxide, or silicon nitride can be used as physical masks on the chip. This distinction will become more apparent as this chapter progresses.

A photomask is constructed with chromium (chrome) covered quartz glass. A UV light source is used to expose the photoresist. Figure 3.1 illustrates the lithography process. The photomask has chrome where light should be blocked. The UV light floods the mask from the backside and passes through the clear sections of the mask to expose the organic photoresist (PR) that has been coated on the wafer. A developer solvent is then used to dissolve the soluble unexposed photoresist, leaving islands of insoluble exposed photoresist. This is termed a *negative* photoresist. A *positive* resist is initially insoluble, and when exposed to UV becomes soluble. Positive resists provide for higher resolution than negative resists, but are less sensitive to light. As feature sizes become smaller, the photoresist layers have to be made thinner. This is turn makes them less robust and more subject to failure. In turn, this can impact the overall yield of a process and the cost to produce the chip.

The photomask is commonly called a *reticle* and is usually smaller than the wafer, e.g., 2 cm on a side. A *stepper* moves the reticle to successive locations to completely expose the wafer. *Projection printing* is normally used, in which lenses between the reticle and wafer focus the pattern on the wafer surface. Older techniques include *contact printing*, where the mask and wafer are in contact, and *proximity printing*, where the mask and wafer are close but not touching. The original form of both techniques can cause damage to the mask and are limited in feature definition in the range of 1–2 μm. However, newer approaches to proximity printing are being proposed for future processes. The reticle can be the same size as the area to be patterned (1X) or larger. For instance, 2.5X and 5X steppers with optical reduction have been used in the industry.

The wavelength of the light source influences the minimum feature size that can be printed [Schellenberg03]. In the 1980s, mercury lamps with 436 nm or 365 nm wavelengths were used. At the 0.25 μm process generation, excimer lasers with 248 nm (deep ultraviolet) were adopted and have been used down to the 180 nm node. Currently 193 nm

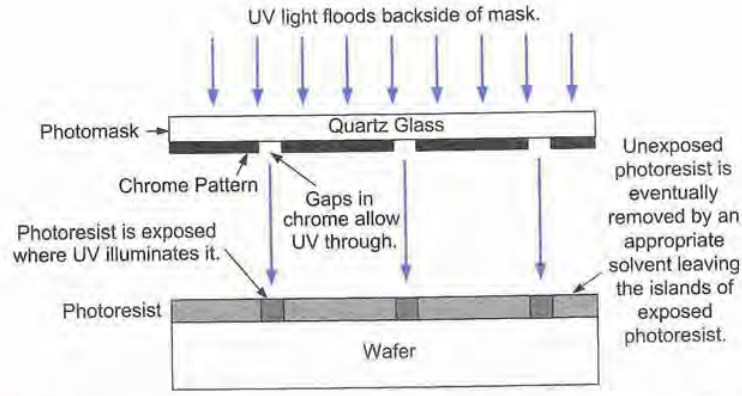


FIG 3.1 Photomasking with a negative resist (lens system between mask and wafer omitted to improve clarity and avoid diffracting the reader ☺)

argon-fluoride lasers are used for the *critical layers* down to the 90 nm node. The critical layers are those that define the device behavior. An example would be the gate (polysilicon), source/drain (diffusion), first metal, and contact masks. The ASML TWINSCAN XT:1250 step and scan lithography tool uses a 193 nm laser to expose up to 150 wafers per hour. It has an alignment tolerance of 8 nm and can process features down to 70 nm. 157 nm fluorine lasers are in development [Nowak02, Miller00] for processes below 90 nm. In the future, 13.4 nm extreme ultraviolet (EUV) light sources may be used, but at present these sources are costly and require prohibitively expensive reflective optics. Each successive UV stepper is more expensive and the throughput of the stepper may decrease. This is just another contributory issue to the spiraling cost of chip manufacturing. The cost of masks is also skyrocketing, forcing chip designers to amortize design and mask expenses across the largest volume possible. This theme will be reinforced in Chapter 8.

Wavelengths comparable to or greater than the feature size cause distortion in the patterns exposed on the photoresist. *Resolution enhancement techniques* precompensate for this distortion so the desired patterns are obtained. These techniques involve modifying the amplitude, phase, or direction of the incoming light. The ends of a line in a layout receive less light than the center, causing nonuniform exposure. *Optical proximity correction* (OPC) makes small changes to the patterns on the masks to compensate for these local distortions. Multiple lines running in parallel on a mask behave as a diffraction grating. *Phase shift masks* (PSM) vary the thickness of the mask to change the phase such that light from adjacent lines are out of phase and cancel where no light is desired. *Off-axis illumination* can also improve contrast for certain types of dense, repetitive patterns. Using these techniques, the resolution can be extended to one-eighth the wavelength of the light.

3.2.4 Well and Channel Formation

Varying proportions of donor and acceptor impurities can be achieved using *epitaxy*, *deposition*, or *implantation*. Epitaxy involves growing a single-crystal film on the silicon surface (which is already a single crystal) by subjecting the silicon wafer surface to an elevated temperature and a source of dopant material.

Epitaxy can be used to produce a layer of silicon with fewer defects than the native wafer surface and also can help prevent latchup (see Section 4.8.5). Foundries usually provide a choice of *epi* (with epitaxial layer) or non-epi wafers. Microprocessor designers usually prefer to use epi wafers for device uniformity of performance.

Deposition involves placing dopant material onto the silicon surface and then driving it into the bulk using a thermal diffusion step. This can be used to build deep junctions. A step called *chemical vapor deposition* (CVD) can be used for the deposition. As its name suggests, CVD occurs when heated gases react in the vicinity of the wafer and produce a product that is deposited on the silicon surface. CVD is also used to lay down thin films of material later in the CMOS process.

Ion implantation involves subjecting the silicon substrate to highly energized donor or acceptor atoms. When these atoms impinge on the silicon surface, they travel below the surface of the silicon, forming regions with varying doping concentrations. At elevated temperature (> 800° C), diffusion occurs between silicon regions having different densities of impurities, with impurities tending to diffuse from areas of high concentration to areas of low concentration. Therefore, it is important to keep the remaining process steps at as low a temperature as possible once the doped areas have been put into place. However, a high-temperature annealing step is often performed after ion implantation to redistribute dopants more uniformly. Ion implantation is the standard well and source/drain implant method used today.

The first step in most CMOS processes is to define the well regions. In a triple-well process, a deep n-well is first driven into the p-type substrate, usually using high-energy (MeV—Mega electron volt levels) ion implantation as opposed to a thermally diffused operation. This avoids the thermal cycling (i.e., the wafers do not have to be raised significantly in temperature), which improves throughput and reliability. A 2–3 MeV implantation can yield a 2.5–3.5 μm deep n-well. Such a well has a peak dopant concentration just under the surface and for this reason is called a *retrograde* well. This can enhance device performance by providing improved latchup characteristics and reduced susceptibility to vertical punch-through. A thick (3.5–5.5 μm) resist has to be used to block the high energy implantation where no well should be formed. Thick resists and deep implants necessarily lead to fairly coarse feature dimensions for wells, compared to the minimum feature size. Shallower n-well and p-well regions are then implanted. After the wells have been formed, the doping levels can be adjusted (called a *threshold implant*) to set the desired threshold voltages for both nMOS and pMOS transistors. For a given gate and substrate material, the threshold voltage (V_t) depends on the doping level in the substrate (N_A), the oxide thickness (t_{ox}), and the surface state charge (Q_{fc}). The implant can affect both N_A and Q_{fc} and hence V_t. Figure 3.2 shows a typical triple-well structure. As

discussed, the nMOS transistors are situated in the p-well located in the deep n-well. The pMOS transistors are located in the shallow (normal) n-well. The figure shows the cross-section of an inverter.

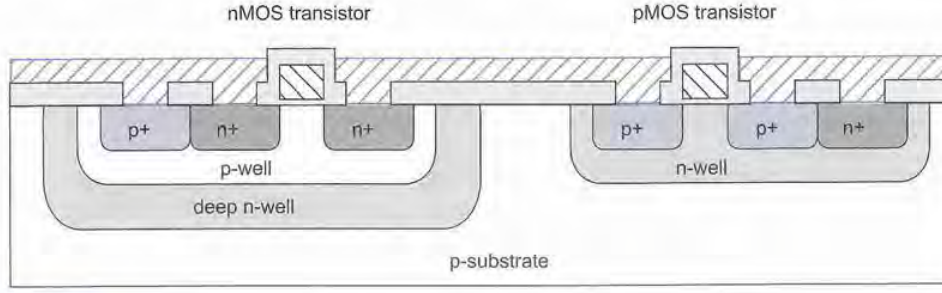


FIG 3.2 Well structure in triple-well process

3.2.5 Silicon Dioxide (SiO_2)

Many of the structures and manufacturing techniques used to make silicon integrated circuits rely on the properties of SiO_2. Therefore, reliable manufacture of SiO_2 is extremely important. In fact, unlike competing materials, silicon has dominated the industry because it has an easily processable oxide (i.e., it can be grown and etched). Various thicknesses of SiO_2 may be required, depending on the particular process. Thin oxides are required for transistor gates; thicker oxides might be required for higher voltage devices, while even thicker oxide layers might be required to ensure that transistors are not formed unintentionally in the silicon beneath polysilicon wires (see the next section).

Oxidation of silicon is achieved by heating silicon wafers in an oxidizing atmosphere. Some common approaches are:

- *Wet oxidation*—when the oxidizing atmosphere contains water vapor. The temperature is usually between 900° C and 1000° C. This is also called *pyrogenic* oxidation when a 2:1 mixture of hydrogen and oxygen is used. Wet oxidation is a rapid process.
- *Dry oxidation*—when the oxidizing atmosphere is pure oxygen. Temperatures are in the region of 1200° C to achieve an acceptable growth rate. Dry oxidation forms a better quality oxide than wet oxidation. It is used to form thin, highly controlled gate oxides, while wet oxidation may be used to form thick field oxides.
- *Atomic layer deposition (ALD)*—a process in which a thin chemical layer (material *A*) is attached to a surface and then a chemical (material *B*) is introduced to produce a thin layer of the required layer (i.e., SiO_2—this can also be used for other various dielectrics and metals). The process is then repeated and the required layer is built up layer by layer. [George96, Klaus98]. Currently, this is an emerging R&D process.

The oxidation process normally consumes silicon (deposition or ALD does not). Since SiO_2 has approximately twice the volume of silicon, the SiO_2 layer grows almost equally in both vertical directions. Thus, after processing, the SiO_2 projects above and below the original unoxidized silicon surface.

3.2.6 Isolation

Although not immediately obvious, individual devices in a CMOS process need to be isolated from one another so that they do not have unexpected interactions. The source and drain of transistors form reverse-biased p–n junctions with the substrate or well, isolating them from their neighbors. Next, the formation of any parasitic MOS channels must be prevented. This is commonly achieved using a thin *gate oxide* for transistors and a much thicker *field oxide* elsewhere. The thicker oxide increases the threshold voltage to a value above the supply voltage and so prevents a channel from forming in the substrate unless there is an overvoltage condition (actually, these *field devices* are used for I/O protection). In addition to using the thick oxide, the substrate in areas where transistors are not required can be further implanted with dopants to create a *channel-stop* diffusion. The implant increases the impurity concentration in the substrate, which in turn raises the threshold voltage and prevents inversion of an unwanted channel. In early metal gate processes that had a uniform (thin) oxide layer, the channel-stop diffusion surrounded each transistor and was the only method of isolating transistors.

Historically, *LOCOS* or Local Oxidation of Silicon was used to produce varying oxide thicknesses. A common problem with LOCOS-based processes was the transition between thick and thin oxide, which extended some distance laterally because of the way the oxide was grown; this in turn limited the packing density of transistors. The isolation step that is used to achieve isolation between devices in processes at and below the 180 nm node is to form insulating trenches of SiO_2 that surround active areas. Typical trenches in a 90 nm process can be 140 nm wide by 400 nm deep [Kim02]. This is called *shallow trench isolation* (STI). (There are also process options with deep trenches.) STI (Figure 3.3) starts with a *pad oxide* and a silicon nitride layer, which act as the masking layers. Openings in the pad oxide are then used to etch into the well or substrate region (this process can also be used for source/drain diffusion). A liner oxide is then grown to cover the exposed silicon. The trenches are filled with SiO_2 using CVD that does not consume the underlying silicon. The pad oxide and nitride are removed and a *Chemical Mechanical Polishing* (CMP) step is used to planarize the structure. CMP, as its name suggests, combines a mechanical grinding action in which the rotating wafer is contacted by a stationary polishing head while an abrasive mixture is applied. The mixture also reacts chemically with the surface to aid in the polishing action. CMP is used to achieve flat surfaces, which are of central importance in modern processes with many layers.

From the designer's perspective, the presence of a deep n-well and/or trench isolation makes it easier to isolate noise-sensitive (analog or memory) portions of a chip from digital sections. Trench isolation also permits nMOS and pMOS transistors to be placed closer together because the isolation provides a higher source/drain *breakdown voltage*—

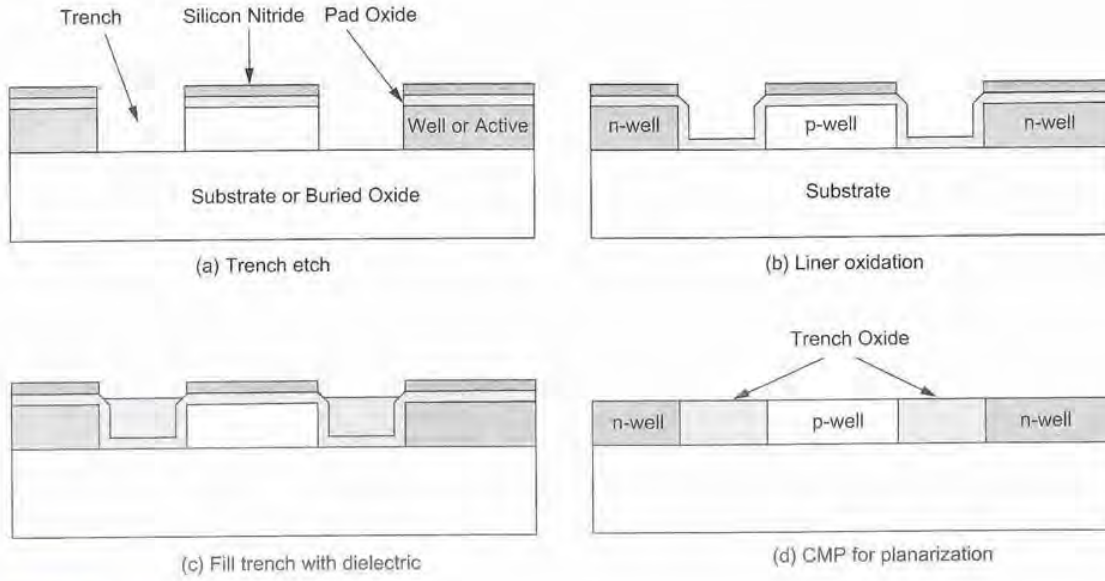


FIG 3.3 Shallow trench isolation

the voltage that a source or drain diodes start to conduct in the reverse-biased condition. The breakdown voltage must exceed the supply voltage (so junctions do not break down during normal operation) and is determined by the junction dimensions and doping levels of the junction formed. Deeper trenches increase the breakdown voltage.

Wells are defined by separate masks. In the case of a twin-well process, only one mask need be defined because the other well is by definition its complement. Triple-well processes have to define at least two masks, one for the deep well and the other for either n-well or p-well.

3.2.7 Gate Oxide

The next step in the process is to form the gate oxide for the transistors. As mentioned, this is most commonly in the form of *silicon dioxide* (SiO_2).

In the case of STI-defined source/drain regions, the gate oxide is grown on top of the planarized structure that occurs at the stage shown in Figure 3.3(d). This is shown in Figure 3.4. The oxide structure is called the *gate stack*. This term arises because current processes seldom use a pure SiO_2 gate oxide, but prefer to produce a stack that consists of a few atomic layers, each 3–4 Å thick, of SiO_2 for reliability, overlaid with a few layers of an *oxynitrided* oxide (one with nitrogen added). The presence of the nitrogen increases the dielectric constant, which decreases the *effective oxide thickness* (EOT); this means that for a given oxide thickness, it performs like a thinner oxide. Being able to use a thicker oxide improves the robustness of the process. This concept is revisited in Section 3.4.1.

Many processes in the 180 nm generation and beyond provide at least two oxide thicknesses (thin for logic transistors and thicker for I/O transistors that must withstand higher voltages). Some processes offer more than one oxide for logic transistors to permit tradeoffs between speed and gate leakage current, as will be discussed in Section 3.4.1. At the 65 nm node, the effective thickness of the thin gate oxide is of the order of 1.5 nm or 15 Å.

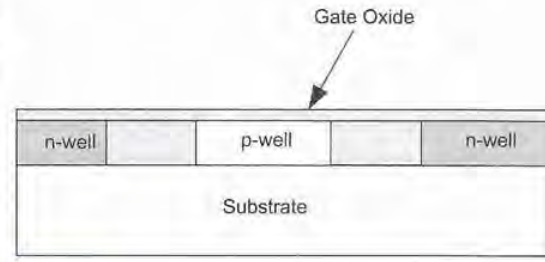


FIG 3.4 Gate oxide formation

3.2.8 Gate and Source/Drain Formation

As a historical note, early metal-gate processes first diffused source and drain regions, and then formed a metal gate. If the gate was misaligned, it could fail to cover the entire channel and lead to a transistor that never turned ON. To prevent this, the metal gate had to overhang the source and drain by more than the alignment tolerance of the process. This created large parasitic gate-to-source and gate-to-drain overlap capacitances that degraded switching speeds.

When silicon is deposited on SiO_2 or other surfaces without crystal orientation, it forms polycrystalline silicon, commonly called *polysilicon* or simply *poly*. An annealing process is used to control the size of the single crystal domains and to improve the quality of the polysilicon. Undoped polysilicon has high resistivity. The resistance can be reduced by implanting it with dopants and/or combining it with a refractory metal. The polysilicon gate serves as a mask to allow precise alignment of the source and drain with the gate. This process is called a *self-aligned polysilicon gate* process. Aluminum could not be used because it would melt during formation of the source and drain.

The steps to define the gate, source, and drain in a self-aligned polysilicon gate are as follows:

- Grow gate oxide wherever transistors are required (area = source + drain + gate)—elsewhere there will be thick oxide (Figure 3.5(a))
- Deposit polysilicon on chip (Figure 3.5(b))
- Pattern polysilicon (both gates and interconnect) (Figure 3.5(c))
- Etch exposed gate oxide—that is, the area of gate oxide that was not covered by polysilicon; at this stage, the chip has windows down to the well or substrate wherever a source/drain diffusion is required (Figure 3.5(d))
- Implant pMOS and nMOS source/drain regions (Figure 3.5(e))

The source/drain implant is relatively low, typically in the range 10^{18}–10^{20} cm^{-3} of impurity atoms. Such a *lightly doped drain* (LDD) structure reduces the electric field at the drain junction (the junction with the highest voltage), which improves the immunity of the device to hot electron damage (see Section 4.8.4). The LDD implants are shallow and lightly doped, so they exhibit low capacitance but high resistance. This reduces device performance somewhat because of the resistance in series with the transistor. Consequently,

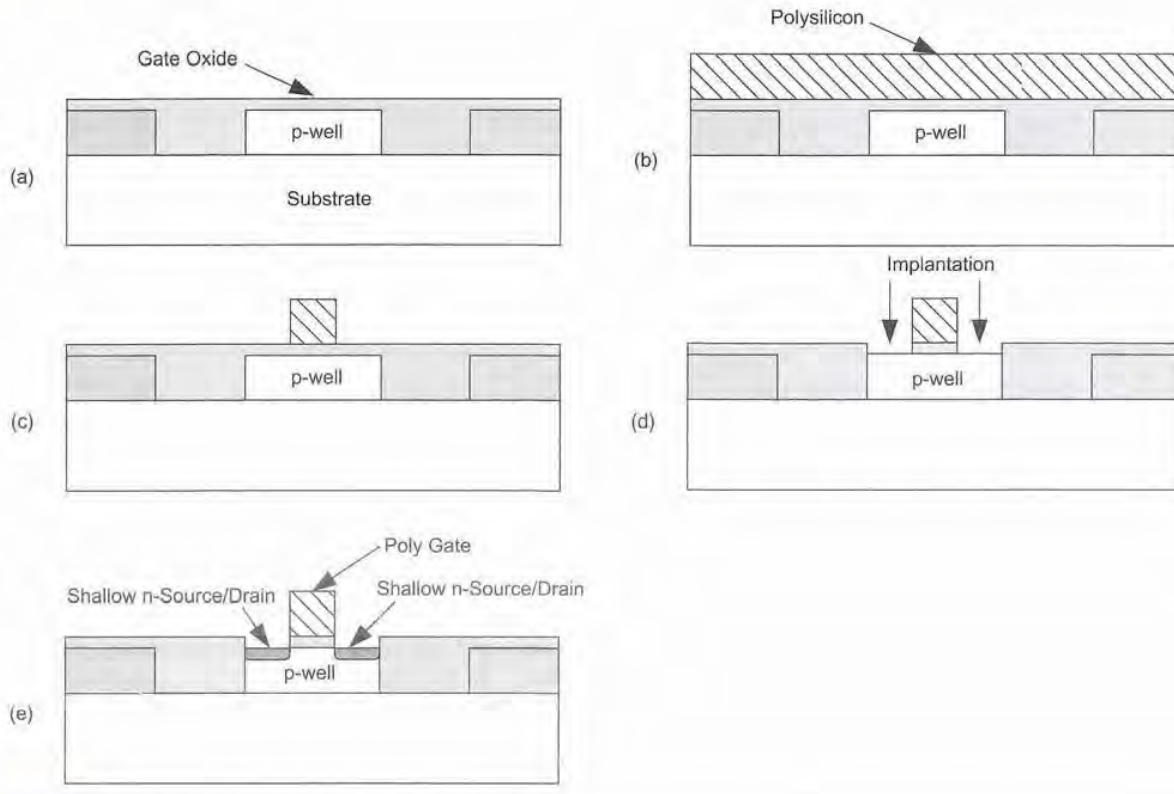


FIG 3.5 Gate and shallow source/drain definition

deeper, more heavily doped source/drain implants are needed to provide devices that combine hot electron suppression with low source/drain resistance. A silicon nitride (Si_3N_4) *spacer* along the edge of the gate serves as a mask to define the location of this deeper diffusion, as shown in Figure 3.6(a). For in-depth coverage of various LDD structures see [Ziegler02].

As mentioned, the polysilicon gate and source/drain diffusion have high resistance due to the resistivity of silicon and their extremely small dimensions. Modern processes form a surface layer of a refractory metal on the silicon to reduce the resistance. A *refractory metal* is one with a high melting point that will not be damaged during subsequent processing. Tantalum, molybdenum, titanium, or cobalt are commonly used. The metal is deposited on the silicon (specifically on the gate polysilicon and/or source/drain regions). A layer of *silicide* is formed when the two substances react at elevated temperatures. In a

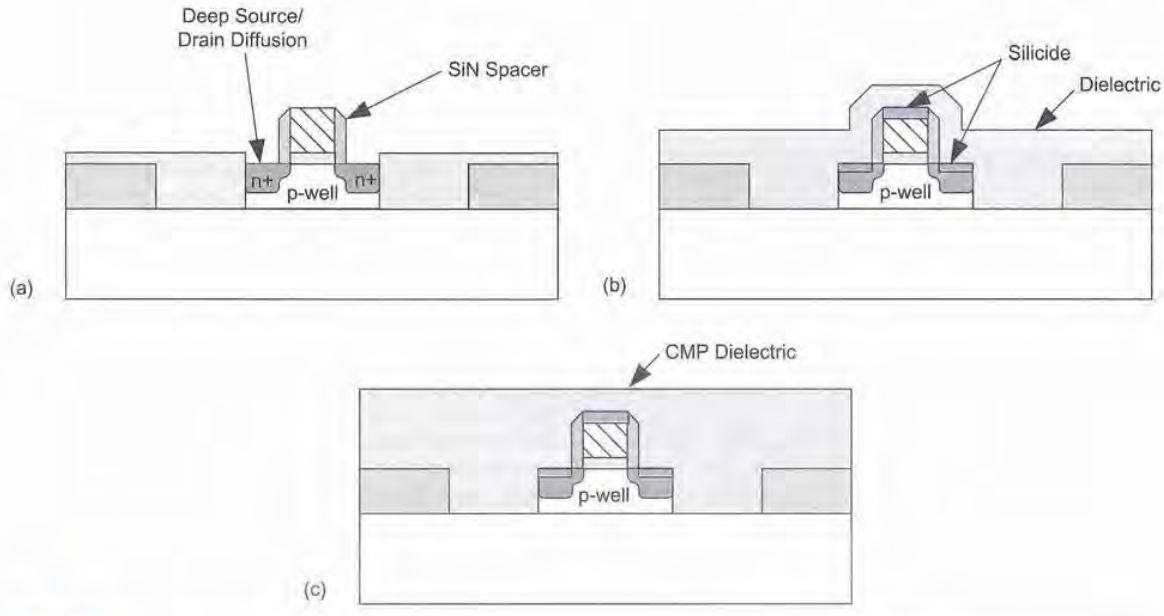


FIG 3.6 Lightly Doped Drain (LDD) structure

polycide process, only the gate polysilicon is silicided. In a silicide process (usually implemented as a self-aligned silicidization—from whence comes the synonymous term *salicide*) both gate polysilicon and source/drain regions are silicided. This process lowers the resistance of the polysilicon interconnect and/or the source and drain.

Figure 3.6(b) shows the resultant structure with gate and source/drain regions silicided. In addition, SiO_2 or an alternative dielectric has been used to cover all areas prior to the next processing steps. The figure shows a resulting structure with some vertical topology typical of older processes.

Figure 3.6(c) shows a structure where CMP has been employed. Achieving a very flat finish allows layers to be stacked vertically without incurring the problems of metal having to traverse rapid transitions in surface height (as shown in Figure 3.6(b)), which can lead to breaks and a plethora of design rules that relate to metal edges.

Polysilicon over diffusion normally forms a transistor gate, so a short metal1 wire is necessary to connect a diffusion output node to a polysilicon input. Some processes add a contact region to the process so that the polysilicon layer can directly connect to the diffusion. Such polysilicon wires are called *local interconnect*. Local interconnect offers denser cell layouts, especially in static RAMs.

3.2.9 Contacts and Metallization

Contact cuts are made to source, drain, and gate according to the contact mask. These are holes etched in the dielectric at the end of the source/drain step covered in the previous section. Aluminum (Al) is commonly used for wires but tungsten (W) can be used as a *plug* to fill the contact holes (to alleviate problems of aluminum conforming to small contacts). In some processes, the tungsten can also be used as a local interconnect layer.

Metallization is the process of building wires to connect the devices. As mentioned previously, conventional metallization uses aluminum. Aluminum can be deposited either by *evaporation* or *sputtering*. Evaporation is performed by passing a high electrical current through a thick aluminum wire in a vacuum chamber. Some of the aluminum atoms are vaporized and deposited on the wafer. An improved form of evaporation that suffers less from contamination focuses an electron beam at a container of aluminum to evaporate the metal. Sputtering is achieved by generating a gas plasma by ionizing an inert gas using an RF or DC electric field. The ions are focused on an aluminum target and the plasma dislodges metal atoms, which are then deposited on the wafer.

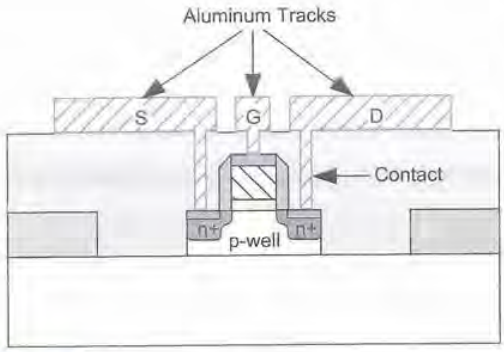


FIG 3.7 Aluminum metallization

Wet or *dry etching* can be used to remove unwanted metal. *Pirhana solution* is a 3:1 to 5:1 mix of sulphuric acid and hydrogen peroxide that is used to clean wafers of organic and metal contaminants or photoresist after metal patterning. Plasma etching is a dry etch process with fluorine or chlorine gas used for metallization steps. The plasma charges the etch gas ions, which are attracted to the appropriately charged silicon surface. Very sharp etch profiles can be achieved using plasma etching. The result of the contact and metallization patterning steps is shown in Figure 3.7.

Subsequent intermetal vias and metallization are then applied. Some processes attempt to use a uniform metallization scheme from at least level 2 to n–1, where n is the top level of metal. The top level is normally a thicker layer for use in power distribution and as such has relaxed width and spacing constraints. Other processes use successively thicker and wider metal, moving from lower to upper layers, as will be explored in Section 4.5. Such a metallization cross-section is shown in Figure 3.8.

3.2.10 Passivation

The final processing step is to add a protective glass layer called *passivation* or *overglass* that prevents the ingress of contaminants. Openings in the passivation layer allow connection to I/O pads and test probe points if needed. After passivation, further steps can be performed such as *bumping*, which allows the chip to be directly connected to a circuit board using plated solder bumps in the pad openings.

3.2.11 Metrology

Metrology is the science of measuring. Everything that is built in a semiconductor process has to be measured to give feedback to the manufacturing process. This ranges from simple optical measurements of line widths to advanced techniques to measure thicknesses of thin films and defects such as voids in copper interconnect. A natural requirement exists for *in situ* real-time measurements so that the manufacturing process can be controlled in a direct feedback manner.

Optical microscopes are used to observe large structures and defects, but are no longer adequate for structures smaller than the wavelength of visible light (~0.5 μm). Scanning electron microscopy (SEM) is used to observe very small features. An SEM raster scans a structure under observation and observes secondary electron emission to produce an image of the surface of the structure. Energy Dispersive Spectroscopy (EDX) bombards a circuit with electrons causing X-ray emission. This can be used for imaging as well. A Transmission Electron Microscope (TEM), which observes the results of passing electrons through a sample (rather than bouncing them off the sample), is sometimes also used to measure structures.

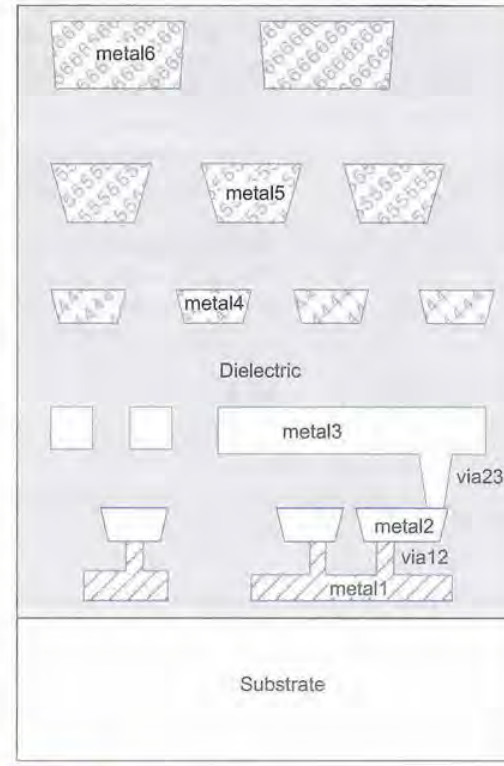


FIG 3.8 Typical metallization cross-section

3.3 Layout Design Rules

Layout rules, also referred to as *design rules* or *ground rules* (IBM), were introduced in Chapter 1 and can be considered a prescription for preparing the photomasks that are used in the fabrication of integrated circuits. The main objective of the layout rules is to build reliably functional circuits in as small an area as possible. In general, design rules represent a compromise between performance and yield. The more conservative the rules are, the more likely it is that the circuit will function. However, the more aggressive the rules are, the greater the opportunity for improvements in circuit performance. This improvement may be the at the expense of yield.

Design rules specify to the designer certain geometric constraints on the layout artwork so that the patterns on the processed wafer will preserve the topology and geometry of the designs. It is important to note that design rules do not represent some hard boundary between correct and incorrect fabrication. Rather, they represent a tolerance that ensures very high probability of correct fabrication and subsequent operation. For example, you may find that a layout that violates design rules can still function correctly and vice

versa. Nevertheless, any significant or frequent departure (*design rule waiver*) from design rules will seriously prejudice the success of a design.

Chapter 1 described a version of design rules based on the MOSIS CMOS Scalable Rules. The MOSIS rules are expressed in terms of λ. These rules allow some degree of scaling between processes, as in principle, you only need to reduce the value of λ and the designs will be valid in the next process down in size. Unfortunately, history has shown that processes rarely shrink uniformly. Thus, industry usually uses the actual micron design rules for layouts. At this point in time, custom layout is usually constrained to a number of often-used standard cells or memories, where the effort expended is amortized over many instances. Only for extremely high-volume chips is the cost savings of a smaller full-custom layout worth the labor cost of that layout.

The rules are defined in terms of *feature sizes* (widths), *separations*, and *overlaps*.

3.3.1 Design Rule Background

We begin by examining the reasons for the most important design rules.

3.3.1.1 Well Rules The n-well is usually a deeper implant (especially a deep n-well) than the transistor source/drain implants, and therefore, it is necessary for the outside dimension to provide sufficient clearance between the n-well edges and the adjacent n^+ diffusions. The inside clearance is determined by the transition of the field oxide across the well boundary. Processes that use STI may permit zero inside clearance. In older LOCOS processes, problems such as the *bird's beak* effect (the lateral space required to transition from thick to thin oxide) usually prevent this. Because the n-well sheet resistance can be several KΩ per square, it is necessary to thoroughly ground the well. This will prevent excessive voltage drops due to well currents. Guidelines on well and substrate taps are given in Section 4.8.5. Where wells are connected to different potentials (say in analog circuits), the spacing rules may differ from *equipotential* wells (all wells at the same voltage—the normal case in digital logic).

Mask Summary: The masks encountered for well specification may include n-well, p-well, and deep n-well. These are used to specify where the various wells are to be placed. Often only one well is specified in a twin-well process (i.e., n-well) and by default the p-well is in areas where the n-well isn't (i.e., p-well equals the logical NOT of the n-well).

3.3.1.2 Transistor Rules CMOS transistors are generally defined by at least four physical masks. These are active (also called diffusion, diff, or thinox), n-select (also called n-implant, nimp, or nplus), p-select (also called p-implant, pimp, or pplus) and polysilicon (also called poly or polyg). The active mask defines all areas where either n- or p-type diffusion is to be placed *or* where the gates of transistors are to be placed. The gates of transistors are defined by the logical AND of the polysilicon mask and the active mask, i.e., where polysilicon crosses diffusion. The select layers define what type of diffusion is required. n-select surrounds active regions where n-type diffusion is required. p-select surrounds areas where p-type diffusion is required. n-diffusion areas inside p-well regions define

nMOS transistors (or n-diffusion wires). n-diffusion areas inside n-well regions define n-well contacts. p-diffusion areas inside n-wells define pMOS transistors (or p-diffusion wires). p-diffusion areas inside p-wells define substrate contacts (or p-well contacts). Frequently, design systems will only define n-diffusion (ndiff) and p-diffusion (pdiff) to reduce the complexity of the process. The appropriate selects are generated automatically. That is, ndiff will be converted automatically into active with an overlapping rectangle or polygon of n-select.

It is essential for the poly to be completely cross active; otherwise the transistor that has been created will be shorted by a diffusion path between source and drain. Hence, poly is required to extend beyond the edges of the active area. This is often termed the *gate extension*. Active must extend beyond the poly gate so that diffused source and drain regions exist to carry charge into and out of the channel. Poly and active regions that should not form a transistor must be kept separated; this results in a spacing rule from active to polysilicon.

Figure 3.9(a) shows the mask construction for the final structures that appear in Figure 3.9(b).

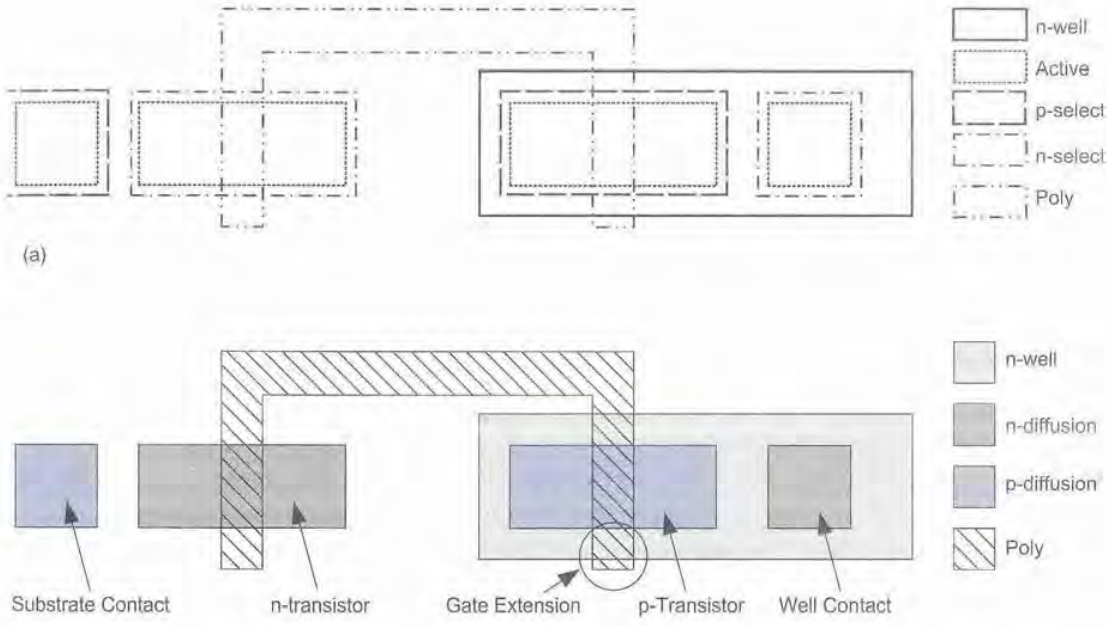


FIG 3.9 CMOS n-well process transistor and well/substrate contact construction

Mask Summary: The basic masks (in addition to well masks) used to define transistors, diffusion interconnect (possibly resistors), and gate interconnect are active, n-select, p-select, and polysilicon. These may be called different names in some processes. Sometimes n-diffusion (ndiff) and p-diffusion (pdiff) masks are used to alleviate designer confusion.

3.3.1.3 Contact Rules There are several generally available contacts:

- Metal to p-active (p-diffusion)
- Metal to n-active (n-diffusion)
- Metal to polysilicon
- Metal to well or substrate

Depending on the process, other contacts such as *buried* polysilicon-active contacts may be allowed for local interconnect.

Because the substrate is divided into well regions, each isolated well must be tied to the appropriate supply voltage; that is, the n-well must be tied to V_{DD} and the substrate or p-well must be tied to GND with well or substrate contacts. As mentioned in Section 1.5.1, metal makes a poor connection to the lightly doped substrate or well. Hence, a heavily doped active region is placed beneath the contact. A *split* or *merged* contact is equivalent to two adjacent contacts to n-active and p-active strapped together with metal. This structure is used to tie transistor sources to the substrate or n-well and simultaneously to GND or V_{DD}, as shown at the source of the n-transistor in Figure 3.10. A split contact is shown, which is consistent with modern processes that usually require uniform contact sizes to achieve well-defined etching characteristics. Merged contact structures in older processes may have used an elongated contact rectangle.

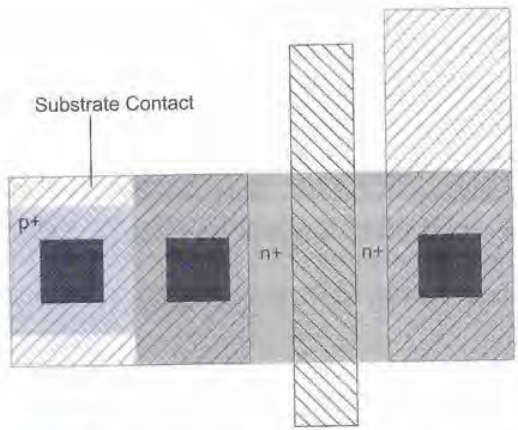


FIG 3.10 Substrate contact

Whenever possible, use more than one contact at each connection. This significantly improves yield in many processes because the connection is still made if one of the contacts is malformed.

Mask Summary: The only mask involved with contacts to active or poly is the contact mask, commonly called CONT. Contacts are normally of uniform size.

3.3.1.4 Metal Rules Metal spacing may vary with the width of the metal line (so called fat-metal rules). That is, above some metal wire width, the minimum spacing may be increased. This is due to etch characteristics of small versus large metal wires. There may also be maximum metal width rules. That is, single metal wires cannot be greater than a certain width. If wider wires are desired, they are constructed by paralleling a number of smaller wires and adding checkerboard links to tie the wires together. Additionally, there may be spacing rules that are applied to long, closely spaced parallel metal lines.

Older nonplanarized processes required greater width and spacing on upper-level metal wires (e.g., metal3) to prevent breaks or shorts between adjoining wires caused by the vertical topology of the underlying layers. This is no longer a consideration for modern planarized processes. Nevertheless, width and spacing are still greater for thicker metal layers.

Mask Summary: Metal rules may be complicated by varying spacing dependent on width: As the width increases, the spacing increases. Metal overlap over contact might be zero or nonzero. Guidelines will also exist for electromigration, as discussed in Section 4.8.2.

3.3.1.5 Via Rules Processes may vary in whether they allow *stacked* vias to be placed over polysilicon and diffusion regions. Some processes allow vias to be placed within these areas, but do not allow the vias to straddle the boundary of polysilicon or diffusion. This results from the sudden vertical topology variations that occur at sub-layer boundaries. Most modern planarized processes allow for stacked vias, which reduces the area required to pass from a lower-level metal to a high-level metal.

Mask Summary: Vias are normally of uniform size within a layer. They may increase in size toward the top of a metal stack. For instance, large vias required on power busses are constructed from an array of uniformly sized vias.

3.3.1.6 Other Rules The passivation or overglass layer is a protective layer of SiO_2 (glass) that covers the final chip. Appropriately sized openings are required at pads and any internal test points.

Some additional rules that might be present in some processes are as follows:

- Extension of polysilicon in the direction that metal wires exit a contact.
- Extension of metal end-of-line region beyond a via.
- Differing pMOS and nMOS gate lengths.
- Differing gate poly extensions depending on the device length or the device construction.

3.3.1.7 Summary Whereas earlier processes tended to be process driven and frequently had long and involved design rules, processes have become increasingly "designer friendly" or, more specifically, computer friendly (most of the mask geometries for designs are algorithmically produced). Companies sometimes create "generic" rules that span a number of different CMOS foundries that they might use. Some processes have design guidelines that feature structures to be avoided to ensure good yields. Traditionally, engineers followed yield-improvement cycles to determine the causes of defective chips and modify the layout to avoid the most common systematic failures. Time to market and product lifecycles are now so short that yield improvement is only done for the highest volume parts. It is often better to reimplement a successful product in a new, smaller technology rather than worry about improving the yield on the older, larger process.

3.3.2 Scribe Line and Other Structures

The *scribe line* surrounds the completed chip where it is cut with a diamond saw. The construction of the scribe line varies from manufacturer to manufacturer. It is designed to prevent the ingress of contaminants from the side of the chip (as opposed to the top of the chip, which is protected by the overglass).

Several other structures are included on a mask including the *alignment mark*, *critical dimension* structures, *vernier* structures, and *process check* structures [Hess94]. The mask alignment mark is usually placed by the foundry to align one mask to the next. Critical dimension test structures can be measured after processing to check proper etching of narrow polysilicon or metal lines. Vernier structures are used to judge the alignment between layers. A vernier is a set of closely spaced parallel lines on two layers. Misalignment between the two layers can be judged by the alignment of the two verniers. Test structures such as chains of contacts and vias and test transistors are used to evaluate contact resistance and transistor parameters. Often these structures can be placed along the scribe line so they do not consume useful wafer area.

3.3.3 MOSIS Scalable CMOS Design Rules

Academic designs often use the λ-based scalable CMOS design rules from MOSIS because they are simple and freely available, and they allow designs to easily migrate from one process to another. These advantages come at the expense of being conservative because they must work for all manufacturing processes.

MOSIS actually has three sets of rules: SCMOS, SUBM, and DEEP. The SUBM rules are somewhat more conservative than SCMOS rules. DEEP rules are even more conservative. The more conservative rules allow you to use a slightly smaller value of λ while still satisfying all of the micron design rules for a process. Table 3.1 lists some of the foundry processes MOSIS has offered and the associate value of λ for the different rule sets. For example, the AMI 0.5 μm process can use the SCMOS rules with λ = 0.35 μm or the SUBM rules with λ = 0.30 μm. SUBM rules are a good choice for class projects because they are somewhat easier to use than DEEP (no half-λ rules), while still being compatible with most processes. Some processes offer a second polysilicon layer for floating-gate transistors and poly-insulator-poly capacitors used in analog circuits discussed in Section 3.4.3.

For design rules where the minimum drawn gate length exceeds the feature size, MOSIS applies a *polysilicon bias* to shrink the gates by a uniform amount before masks are made. For example, in the SUBM rules for the AMI 0.5 μm process with λ = 0.3 μm, a bias of −0.1 μm is applied to all polysilicon. Thus, a 2 λ transistor gate is 0.5 μm rather than 0.6 μm and a 4 λ gate is 1.1 μm rather than 1.2 μm. When simulating circuits, be sure to use the biased channel lengths to accurately model the transistor behavior. In SPICE, the XL parameter is added to the specified transistor length to find the actual length. For example, a SPICE deck could specify λ = 0.3um for each transistor and include XL = −0.1um in the model file to indicate a biased length of 0.5 μm.

Table 3.1 MOSIS design rule options

Vendor	Feature Size (μm)	Interconnect Layers	Stacked Vias	SCMOS	SUBM	DEEP
Orbit	2.0	2 metal	No	λ = 1.0 μm		
AMI	1.5	2 metal, 2 poly	No	λ = 0.80 μm	λ = 0.80 μm	
AMI	0.5	3 metal, 1–2 poly	Yes	λ = 0.35 μm	λ = 0.30 μm	
TSMC	0.35	4 metal, 1–2 poly	Yes		λ = 0.20 μm	
TSMC	0.25	5 metal	Yes		λ = 0.15 μm	λ = 0.12 μm
TSMC	0.18	6 metal	Yes		λ = 0.10 μm	λ = 0.09 μm

Section 1.5.3 introduced the basic design rules. Table 3.2 lists in more detail the MOSIS design rules that are generally relevant to designers for a process with *N* metal layers. Figure 3.11 illustrates many of these rules and they are shown in color on the inside back cover. See `www.mosis.org` for complete, up-to-date rules with illustrations. Layouts consist of a set of rectangles on various layers such as polysilicon or metal. *Width* is the minimum width of a rectangle on a particular layer. *Spacing* is the minimum spacing between two rectangles on the same or different layers. *Overlap* specifies how much a rectangle must surround another on another layer. Dimensions are all specified in λ except for overglass cuts that do not scale well because they must contact bond wires or probe tips. Select layers are often generated automatically and thus are not shown in the layout. If the active layer satisfies design rules, the select will too.

Contacts and vias must be exactly 2x2 λ. Larger connections are made from arrays of small vias to prevent current crowding at the periphery. The spacing rules of polysilicon or diffusion to arrays of multiple contacts is slightly larger than that to a single contact.

Section 1.6.5 estimated the pitch of lower-level metal to be 8 λ–4 λ for the width, and 4 λ for spacing. Technically, the minimum width and spacing are 3 λ, but the minimum metal contact size is 2x2 λ plus 1 λ surround on each side, for a width of 4 λ. Thus, the pitch for contacted metal lines can be reduced to 7 λ. Moreover, if the lines are drawn at 3 λ and the contacts are staggered so two adjacent lines never have adjacent contacts, the pitch reduces to 6.5 λ. Nevertheless, using a pitch of 8 λ for planning purposes is good practice and leaves a bit of "wiggle room" to solve difficult layout problems.

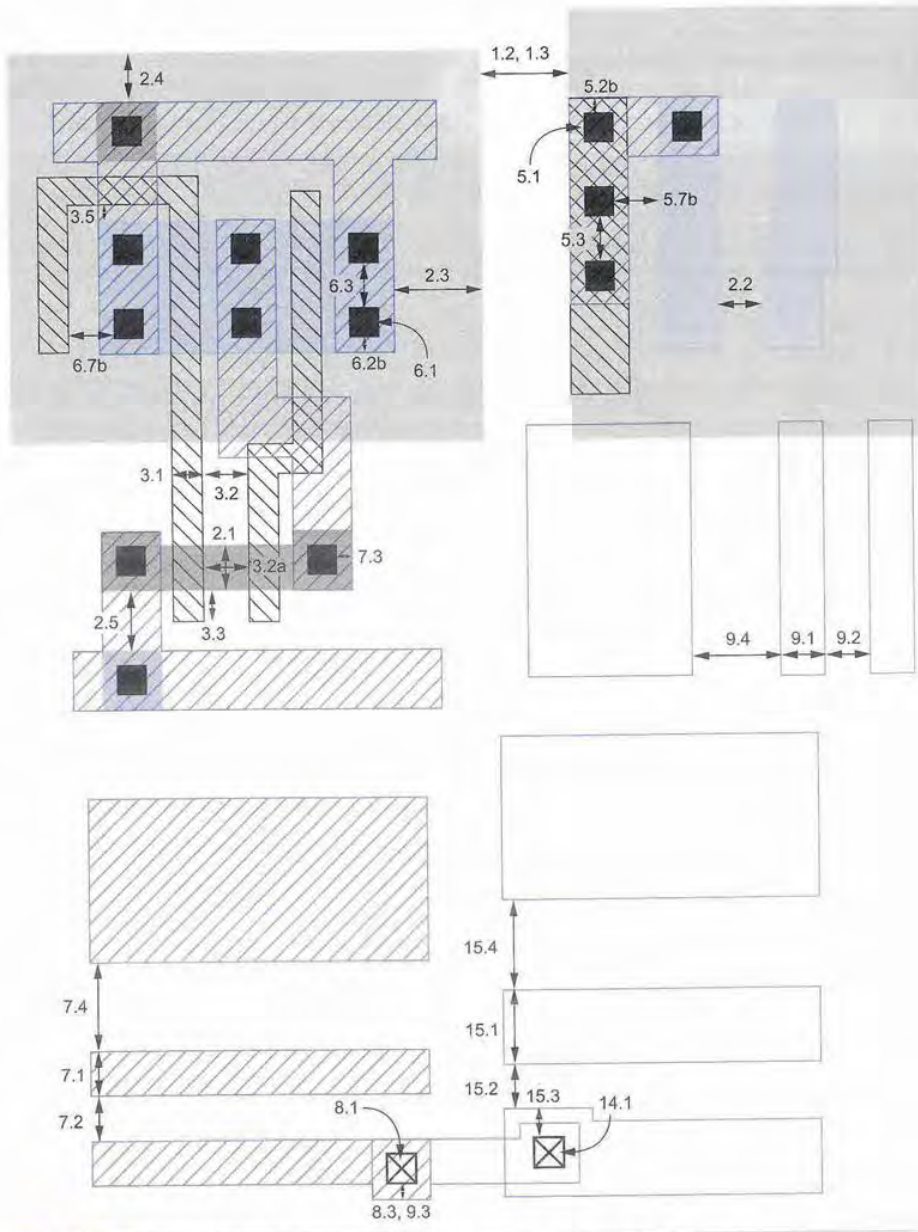


FIG 3.11 Illustrated MOSIS design rules. Color version on inside back cover.

Table 3.2 MOSIS design rules

Layer	Rule	Description	SCMOS	SUBM	DEEP
Well	1.1	Width	10	12	12
	1.2	Spacing to well at different potential	9	18	18
	1.3	Spacing to well at same potential	6	6	6
Active (diffusion)	2.1	Width	3	3	3
	2.2	Spacing to active	3	3	3
	2.3	Source/drain surround by well	5	6	6
	2.4	Substrate/well contact surround by well	3	3	3
	2.5	Spacing to active of opposite type	4	4	4
Poly	3.1	Width	2	2	2
	3.2	Spacing to poly over field oxide	2	3	3
	3.2a	Spacing to poly over active	2	3	4
	3.3	Gate extension beyond active	2	2	2.5
	3.4	Active extension beyond poly	3	3	4
	3.5	Spacing of poly to active	1	1	1
Select	4.1	Spacing from substrate/well contact to gate	3	3	3
	4.2	Overlap of active	2	2	2
	4.3	Overlap of substrate/well contact	1	1	1.5
	4.4	Spacing to select	2	2	4
Contact (to poly or active)	5.1, 6.1	Width (exact)	2x2	2x2	2x2
	5.2b, 6.2b	Overlap by poly or active	1	1	1
	5.3, 6.3	Spacing to contact	2	3	4
	5.4, 6.4	Spacing to gate	2	2	2
	5.5b	Spacing of poly contact to other poly	4	5	5
	5.7b, 6.7b	Spacing to active/poly for multiple poly/ active contacts	3	3	3
	6.8b	Spacing of active contact to poly contact	4	4	4
Metal1	7.1	Width	3	3	3
	7.2	Spacing to metal1	2	3	3
	7.3, 8.3	Overlap of contact or via	1	1	1
	7.4	Spacing to metal for lines wider than 10 λ	4	6	6

Table 3.2 MOSIS design rules (continued)

Layer	Rule	Description	SCMOS	SUBM	DEEP
Via1–Via(*N*-1)	8.1, 14.1, ...	Width (exact)	2x2	2x2	3x3
	8.2, 14.2, ...	Spacing to via on same layer	3	3	3
	8.4	Spacing to contacts (if no stacked vias)	2	2	n/a
	8.5	Spacing of via1 to poly or active edge	2	2	n/a
	14.4	Spacing of via2 to via1 (if no stacked vias)	2	2	n/a
Metal2–Metal(*N*-1)	9.1, ...	Width	3	3	3
	9.2, ...	Spacing to same layer metal	3	3	4
	9.3, ...	Overlap of via	1	1	1
	9.4, ...	Spacing to metal for lines wider than 10 λ	6	6	8
Metal3 (3-layer process)	15.1	Width	6	5	n/a
	15.2	Spacing to metal3	4	3	n/a
	15.3	Overlap of via2	2	2	n/a
	15.4	Spacing to metal for lines wider than 10 λ	8	6	n/a
Metal5 (5-layer process)	26.1	Width	n/a	4	4
	26.2	Spacing to metal5	n/a	4	4
	26.3	Overlap of via4	n/a	1	2
	26.4	Spacing to metal for lines wider than 10 λ	n/a	8	8
Metal6 (6-layer process)	30.1	Width	n/a	5	5
	30.2	Spacing to metal6	n/a	5	5
	30.3	Overlap of via5	n/a	1	2
	30.4	Spacing to metal for lines wider than 10 λ	n/a	10	10
Overglass Cut	10.1	Width of bond pad opening	60 μm		
	10.2	Width of probe pad opening	20 μm		
	10.3	Metal overlap of overglass cut	6 μm		
	10.4	Spacing of pad metal to unrelated metal	30 μm		
	10.5	Spacing of pad metal to active or poly	15 μm		

3.3.4 Micron Design Rules

Table 3.3 lists a set of micron design rules for a hypothetical 90 nm process representing an amalgamation of several real processes. Observe that the rules differ slightly but not immensely from lambda-based rules with λ = 0.05 μm.

Table 3.3 Micron design rules for 90nm process

Layer	Rule	Description	90 nm rule (μm)
Well	1.1	Width	0.75
	1.2	Spacing to well at different potential	1.5
	1.3	Spacing to well at same potential	1.0
Active (diffusion)	2.1	Width	0.15
	2.2	Spacing to active	0.20
	2.3	Source/drain surround by well	0.25
	2.4	Substrate/well contact surround by well	0.25
	2.5	Spacing to active of opposite type	0.30
Poly	3.1	Width	0.09
	3.2	Spacing to poly over field oxide	0.15
	3.2a	Spacing to poly over active	0.15
	3.3	Gate extension beyond active	0.15
	3.4	Active extension beyond poly	0.15
	3.5	Spacing of poly to active	0.10
Select	4.1	Spacing from substrate/well contact to gate	0.25
	4.2	Overlap of active	0.20
	4.3	Overlap of substrate/well contact	0.10
	4.4	Spacing to select	0.30
Contact (to poly or active)	5.1, 6.1	Width (exact)	0.12
	5.2b, 6.2b	Overlap by poly or active	0.01
	5.3, 6.3	Spacing to contact	0.15
	5.4	Spacing to gate	0.10
Metal1	7.1	Width	0.13
	7.2	Spacing to well metal1	0.13
	7.3, 8.3	Overlap of contact or via	0.01
	7.4	Spacing to metal for lines wider than 0.5 μm	0.40
Via1–Via5	8.1, 14.1, ...	Width (exact)	0.13
	8.2, 14.2, ...	Spacing to via on same layer	0.13

(continued)

Table 3.3 Micron design rules for 90nm process (continued)

Layer	Rule	Description	90 nm rule (μm)
Metal2–Metal6	9.1, ...	Width	0.15
	9.2, ...	Spacing to same layer metal	0.15
	9.3, ...	Overlap of via	0.01
	9.4, ...	Spacing to metal for lines wider than 1.0 μm	0.40
Via6		Width	0.20
		Spacing	0.20
Metal7		Width	0.40
		Spacing to metal7	0.40
		Overlap of via6	0.10
		Spacing to metal7 for lines wider than 1.0 μm	0.50

3.4 CMOS Process Enhancements

3.4.1 Transistors

3.4.1.1 Multiple Threshold Voltages and Oxide Thicknesses It has been mentioned that some processes offer multiple threshold voltages and/or oxide thicknesses. Low-threshold transistors deliver more ON current, but also have greater subthreshold leakage. Providing two or more thresholds permits the designer to use low-V_t devices on critical paths and higher-V_t devices elsewhere to limit leakage power. Multiple masks and implantation steps are used to set the various thresholds.

Thin gate oxides also permit more ON current. However, they break down when exposed to the high voltages needed in I/O circuits. Very thin oxides also contribute to large gate leakage currents. Many processes offer a second, thicker oxide for the I/O transistors (see Section 12.4.3). For example, 3.3 V I/O circuits commonly use 0.35 μm channel lengths and 7 nm gate oxides. An intermediate oxide thickness for low-leakage logic circuits can also be useful. Again, multiple masks are used to define the different oxides.

3.4.1.2 Silicon on Insulator A variant of CMOS that has been available for many years and is currently emerging in prominence is Silicon on Insulator (SOI). As the name suggests, this is a process where the transistors are fabricated on an insulator. Two main insulators are used, SiO_2 and sapphire. One major advantage of an insulating substrate is the elimination of the capacitance between the source/drain regions and body, leading to

higher-speed devices. Another major advantage is lower subthreshold leakge. Proponents argue that SOI will be a core technology required to scale devices.

Figure 3.12 shows two common types of SOI. Figure 3.12(a) illustrates a sapphire substrate. In this technology (for example, Peregrine Semiconductor's Ultra Thin Silicon (UTSi), a thin layer of silicon is formed on the sapphire surface. The thin layer of silicon is selectively doped to define different threshold transistors. Gate oxide is grown on top of this and then polysilicon gates are defined. Following this, the nMOS and pMOS transistors are formed by implantation. Figure 3.12(b) shows a silicon-based SOI process. Here, a silicon substrate is used and a *buried oxide* (BOX) is grown on top of the silicon substrate. A thin silicon layer is then grown on top of the buried oxide and this is selectively implanted to form nMOS and pMOS transistor regions. Gate, source, and drain regions are then defined in a similar fashion to a bulk process. Sapphire is optically and RF transparent. As such, it can be of use in optoelectronic areas when merged with III-V based light emitters.

SOI devices and circuits are discussed further in Section 6.7.

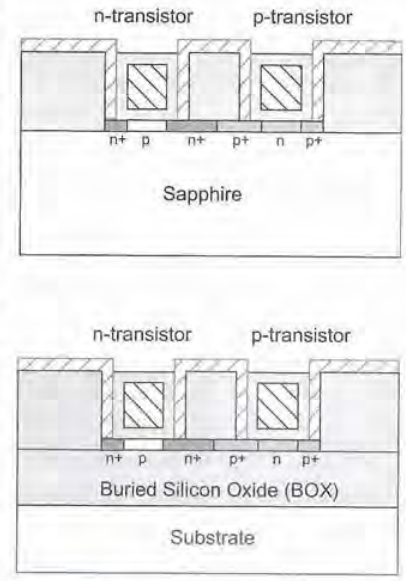


FIG 3.12 SOI types

3.4.1.3 High-k Gate Dielectrics MOS transistors need high gate capacitance to attract charge to the channel. This leads to very thin SiO_2 gate dielectrics. Scaling trends indicate the gate leakage will be unacceptably large in such thin gates. Gates could use thicker dielectrics and hence leak less if a material with a higher dielectric constant were available. Materials such as hafnium oxide HfO_2 (dielectric constant k = 20), zirconium oxide ZrO_2 (k = 23), and silicon nitride Si_3N_4 (k = 6.5–7.5) [Ma98] have been proposed. These are called high-k dielectrics in contrast to SiO_2 with k = 3.9. They are applied using ALD, metallo-organic chemical vapor deposition (MOCVD) or sputtering [Wong02].

3.4.1.4 Low-leakage Transistors Another problem with scaling bulk transistors is the subthreshold leakage from drain to source caused by the inability of the gate to turn off the channel. This can be improved by a gate structure where the gate is placed on two, three, or four sides of the channel. A promising candidate solves the problem by forming a vertical channel and constructing the gate in a pincer-like arrangement. These devices have been given the generic name "*finfets*" because the source/drain region forms fins on the silicon surface [Hisamoto98]. Figure 3.13(a) shows a 3D view of a finfet, while Figure 3.13(b) shows the cross-section and Figure 3.13(c) shows the top view. The gate wraps around three sides of the vertical source/drain fins. The width of the device is defined by the height of the fin, so wide devices are constructed by paralleling fins. Various other device structures have been proposed that even try to wrap the gate around the channel.

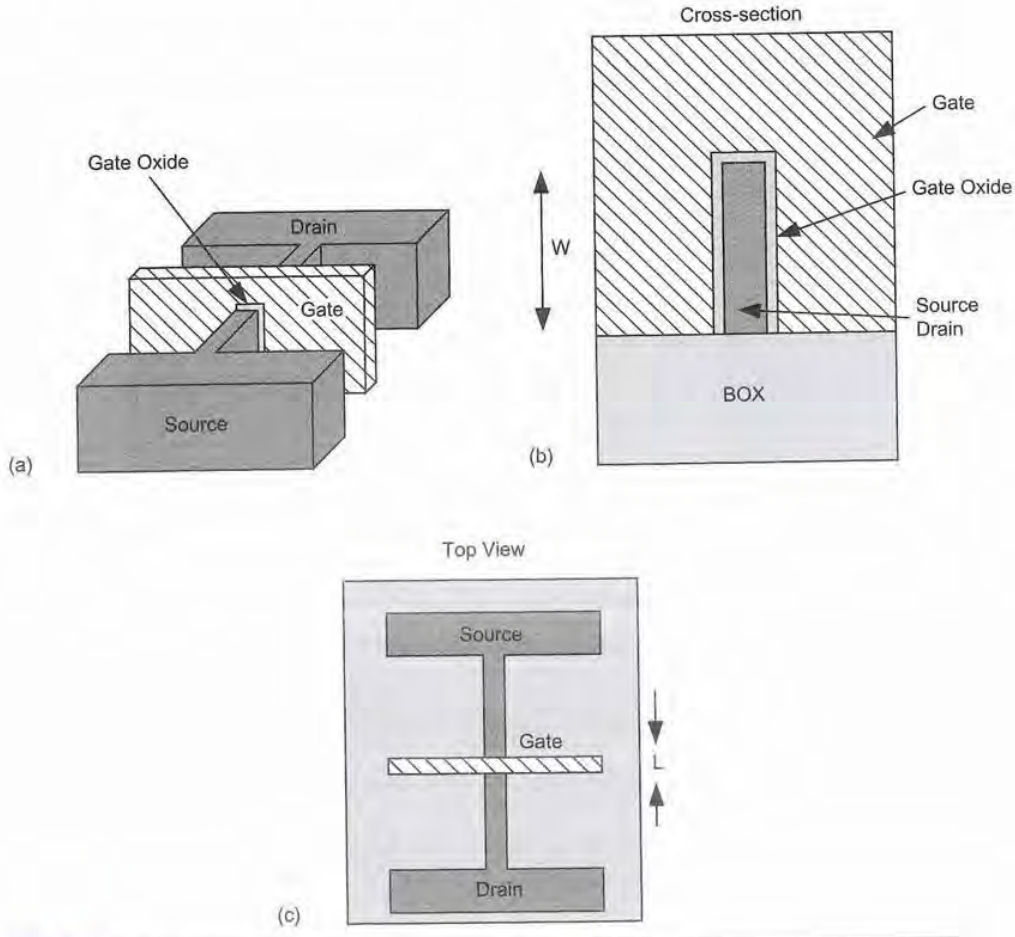


FIG 3.13 Finfet structure

3.4.1.5 Higher Mobility Increasing the mobility (μ) of the semiconductor improves drive current and transistor speed. This has been achieved by using silicon germanium (SiGe) for bipolar transistors in the same process as conventional CMOS devices. A typical SiGe bipolar transistor is shown in Figure 3.14. SiGe transistors can be constructed on conventional CMOS processing by adding a few extra implantation steps. The resulting bipolar transistors have extremely good radio frequency (RF) performance. Advanced SiGe transistors exhibit performance parameters that even III-V compounds such as

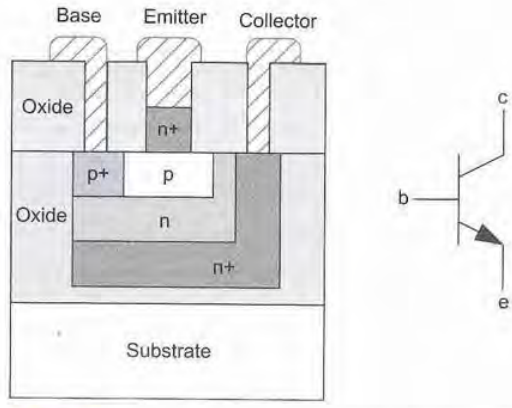


FIG 3.14 SiGe bipolar transistor structure

GaAs and InP find difficult to achieve in production. As a result of the combination of excellent RF performance and high-density digital CMOS, SiGe-based CMOS processes find wide application in communications circuits involving RF and high-speed switching [Hashimoto02, Harame01a, Harame01b].

Silicon germanium can also be used to improve the speed of conventional MOS transistors by creating what is called "*strained silicon*"—silicon into which is implanted germanium atoms that stretch the silicon lattice—as shown in Figure 3.15. This yields an increase in the mobility of the devices over conventional silicon of up to 70% and which corresponds to roughly a 30% increase in performance.

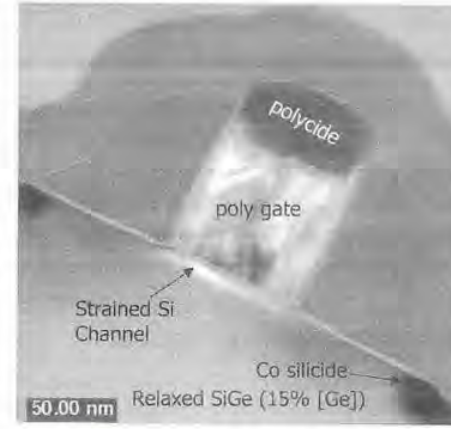


FIG 3.15 IBM strained silicon transistor. Courtesy of International Business Machines Corporation. Unauthorized use not permitted.

3.4.1.6 Plastic Transistors MOS transistors can be fabricated with organic chemicals. These transistors show promise in active matrix displays or flexible electronic paper because the devices can be manufactured from an inexpensive chemical solution [Huitema03]. Figure 3.16 shows the structure of a plastic pMOS transistor. The transistor is built "upside down" with the gold gates and interconnect patterned first on the substrate. Then an organic insulator or silicon nitride is laid down, followed by the gold source and drain connections. Finally, the organic semiconductor (pentacene) is laid down. The mobility of the carriers in the plastic pMOS transistor is about 20 $cm^2/V \cdot s$. This is about one-tenth that of a comparable silicon device. Typical lengths and widths are 5 μm and 400 μm, respectively.

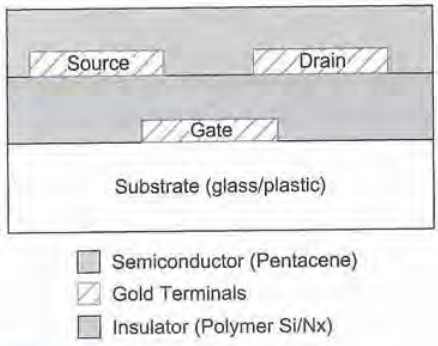


FIG 3.16 Plastic transistors

3.4.1.7 High-voltage Transistors High-voltage MOSFETs can also be integrated onto conventional CMOS processes for switching and high-power applications. Gate oxide thickness and channel length have to be larger than usual to prevent breakdown. Specialized process steps are necessary to achieve very high breakdown voltages.

3.4.2 Interconnect

Interconnect has advanced rapidly. While two or three metal layers were once the norm, CMP has enabled inexpensive processes to include seven or more layers. Copper metal and low-k dielectrics are becoming popular to reduce the resistance and capacitance of these wires.

3.4.2.1 Copper Damascene Process While aluminum has traditionally remained the interconnect metal of choice, a large development effort has centered on using copper as an interconnect metal. This is primarily due to the higher conductivity of copper compared to aluminum. Some challenges of adopting copper include [Merchant01]:

- Copper atoms diffuse into the silicon and dielectrics, destroying transistors.
- The processing required to etch copper wires is tricky.
- Copper oxide forms readily and interferes with good contacts.
- Care has to be taken not to introduce copper into the environment as a pollutant.

Barrier layers have to be used to prevent the copper from entering the silicon surface. A new metallization procedure called the *damascene process* was invented to form this barrier. The process gets its name from the medieval metallurgists of Damascus who crafted fine inlaid swords. In a conventional subtractive aluminum-based metallization step, as we have seen, aluminum is layered on the silicon surface (where vias also have been etched) and then a mask and resist used to define which areas of metal are to be retained. The unneeded metal is etched away. A dielectric (SiO_2 or other) is then placed over the aluminum conductors and the process can be repeated.

A typical copper damascene process is shown in Figure 3.17, which is an adaptation of a dual damascene process flow from Novellus. Figure 3.17(a) shows a barrier layer over the prior metallization layer. This stops the copper from diffusing into the dielectric and silicon. The via dielectric is then laid down (Figure 3.17(b)). A further barrier layer can then be patterned and the line dielectric is layered on top of the structure as shown in Figure 3.17(c). An anti-reflective layer (which helps in the photolithographic process) is added to the top of the sandwich. The two dielectrics are then etched away where the lines and vias are required. A barrier layer such as 10 nm thick Ta or TaN film is then deposited to prevent the copper from diffusing into the dielectrics [Peng02]. As can be seen, a thin layer of the barrier remains at the bottom of the via so the barrier must be conductive. A

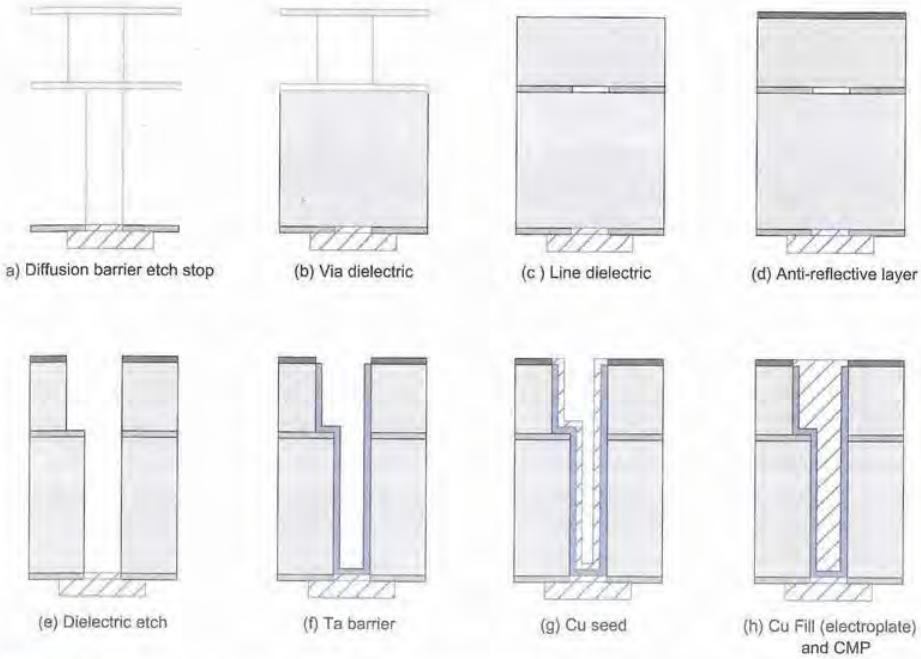


FIG 3.17 Copper dual damascene interconnect processing steps

copper seed layer is then coated over the barrier layer (Figure 3.17(g)). The resulting structure is electroplated full of copper and finally the structure is ground flat with CMP as shown in Figure 3.17(h).

3.4.2.2 Low-k Dielectrics SiO_2 has a dielectric constant of k = 3.9–4.2. Low-k dielectrics between wires are attractive because they decrease the wire capacitance [Brown03]. This reduces both wire delay and power consumption. Adding fluorine to the silicon dioxide creates fluorosilicate glass (FSG) with a dielectric constant of 3.6, widely used in 130 nm processes. Adding carbon to the oxide can reduce the dielectric constant to 2.7–3. Alternatively, porous polymer-based dielectrics can deliver even lower dielectric constants. For example, SiLK, from Dow Chemical, has k = 2.6 and may scale to k = 1.6–2.2 by increasing the porosity. Developing low-k dielectrics that can withstand the high temperatures during processing and the forces applied during CMP is a major challenge.

3.4.3 Circuit Elements

While CMOS transistors provide for almost complete digital functionality, the use of CMOS technology as the mixed signal and RF process of choice have driven the addition

of special process options to enhance the performance of circuit elements required for these purposes.

3.4.3.1 Capacitors In a conventional CMOS process, a capacitor can be constructed using the gate and source/drain of an MOS transistor, a diffusion area (to ground or V_{DD}), or a parallel metal plate capacitor (using stacked metal layers). The MOS capacitor has good capacitance per area but is relatively nonlinear if operated over large voltage ranges. The diffusion capacitor cannot be used for a floating capacitor (but is useful as a bypass capacitor). The metal parallel plate capacitor has low capacitance per area. Normally the aim in using a floating capacitor is to have the highest ratio of desired capacitance value to stray capacitance (to ground normally). The bottom metal plate contributes stray capacitance to ground.

Analog circuits frequently require capacitors in the range of 1 to 10 pF. The first method for doing this was to add a second polysilicon layer so that a *poly-insulator-poly* (PIP) capacitor could be constructed. A thin oxide was placed between the two polysilicon layers to achieve capacitance of approximately 1 fF/μm^2.

The most common capacitor used in CMOS processes today is the MIM or metal-insulator-metal capacitor that is normally placed between metal layers n and $n-1$ (where n is normally the top level metal layer) to minimize the stray capacitance of the bottom plate. A typical insulator is an alumina (Al_2O_3)/tantalum pentoxide (Ta_2O_5) sandwich. These capacitors have capacitances of 1–4 fF/μm^2 and provide very area efficient capacitors. A typical MIM capacitor is shown in Figure 3.18(a).

Another type of capacitor that is possible in scaled processes is the fringe (or fractal) capacitor [Samavati98, Sowiati01], which is composed of interdigitated metal fingers as shown in Figure 3.18(b). The original fractal capacitor [Samavati98] has a more involved layout and was necessary in older processes. Successive metal layers can be ganged to achieve more capacitance. If the upper layers of metal are used, a very linear, high-Q and high valued capacitor with low parasitic capacitance to ground can be constructed without any extra process steps at almost the same values as MIM capacitors. This approach works best at 130 nm and below where the metal lines are very closely spaced and thus have high fringing capacitance.

Integrated capacitors have a voltage and temperature dependence. Foundry design guides should be consulted for these parameters.

3.4.3.2 Resistors In unaugmented processes, resistors can be built from any layer, with the final resistance depending on the resistivity of the layer. Building large resistances in a small area requires layers with high resistivity, particularly polysilicon and diffusion. Diffusion has a large parasitic capacitance to ground, making it unsuitable for high-frequency applications. Polysilicon gates are usually silicided to have low resistivity. The fix for this is to allow for *undoped* high-resistivity polysilicon. This is specified with a mask that blocks the silicide where high-value poly resistors are required. The resistivity can be tuned to around 300–1000 ohms/square, depending on doping levels. Another material used for high-quality resistors is nichrome, although this requires a special processing step.

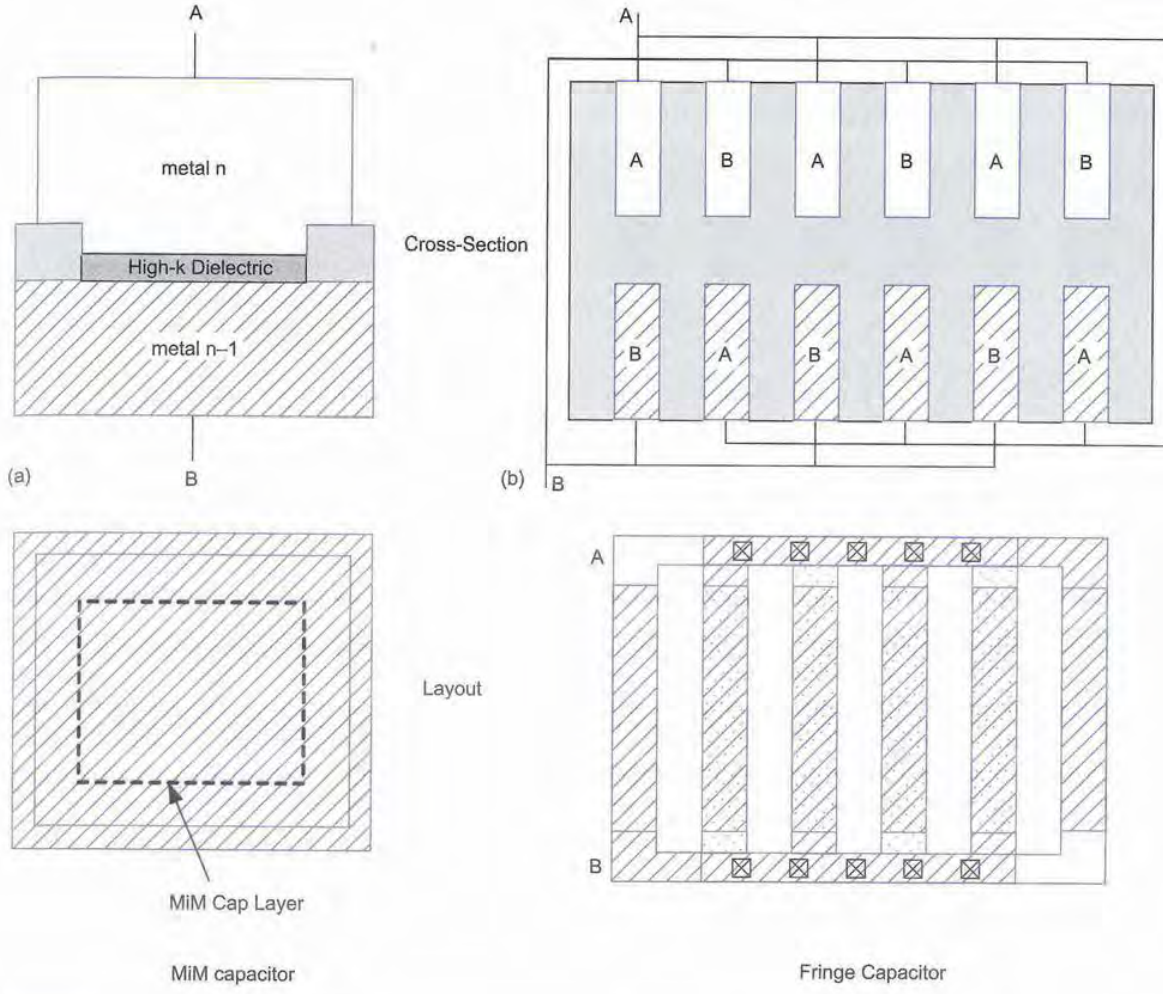


FIG 3.18 MIM and fringe capacitors

A typical resistor layout is shown in Figure 3.19. This geometry is sometimes called a *meander* structure. A number of *unit* resistors have been used so that a variety of matched resistor values can be constructed. For instance, if 20K, 10K, and 15K ohm resistors were required, a unit value of 5K could be used. Then three resistors (as shown) would construct a 15K ohm resistor. The two resistors at the ends are called *dummy resistors* or fingers.

They perform no circuit function, but replicate the proximity effects (such as etch and implant) that the interior resistors see during processing. This helps ensure that all resistors are matched.

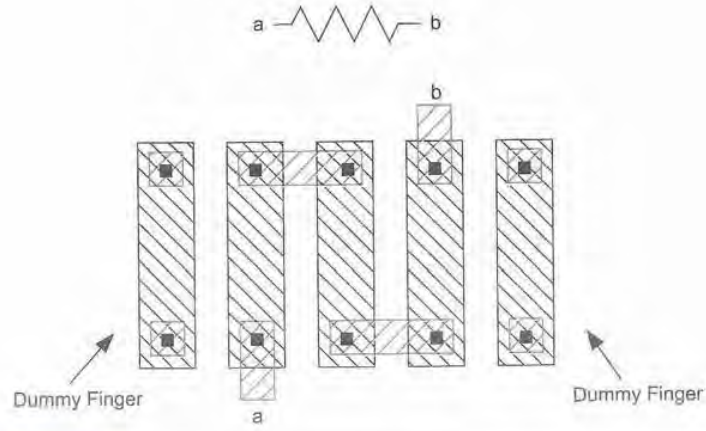


FIG 3.19 Resistor layout

Like integrated capacitors, the various resistor options have temperature and voltage coefficients. Foundry design manuals normally include these values.

3.4.3.3 Inductors The desire to integrate inductors on chips has increased radically with the upsurge in interest in RF circuits. The most common monolithic inductor is the *spiral* inductor, which is a spiral of upper-level metal. A typical inductor is shown in Figure 3.20. As the process is planar, an underpass connection has to be made to complete the inductor. A typical equivalent model is also shown in Figure 3.20. In addition to the required L, there are several parasitic components. Rs is the series resistance of the metal (and contacts) used to form the inductor. Cp is the parallel capacitance to ground due to the area of the metal wires forming the inductor. Cs is the shunt capacitance of the underpass. Finally, Rp is an element that models the loss incurred in the resistive substrate.

Usually, when considering an inductor, the parameters of interest to a designer are its inductance, the Q of the inductor, and the self-resonant frequency. High Q's are sought to create low phase-noise oscillators, narrow filters, and low-loss circuits in general. Q values for typical planar inductors on a bulk process are in the range from 5 to 10.

The number of turns n required to achieve some inductance L if the wire pitch is $P = W + S$ is [Lee98]

$$n \approx \sqrt[3]{\frac{LP}{\mu_0}} \tag{3.1}$$

where $\mu_0 = 1.2 \cdot 10^{-6}$ H/m is the permeability of free space. Higher-quality inductors can also be manufactured using bond wires between I/O pads. The inductance of a wire of length l and radius r is approximately

$$L \approx \frac{\mu_0 l}{2\pi}\left[\ln\frac{2l}{r} - 0.75\right] \qquad (3.2)$$

or about 1 nH/mm for standard 1 mil (25 μm) bond wires.

Reduction in Q occurs because of the resistive loss in the conductors used to build the inductor (Rs) and the eddy current loss in the resistive silicon substrate (Rp). In an effort to increase Q, designers have resorted to removing the substrate below the inductor using MEMS techniques [Yoon02]. Designers have also used bond wires for high Q inductors. The easiest way to improve the Q of monolithic inductors is to increase the thickness of the top level metal. The Q can also be improved by using a patterned ground shield in polysilicon under the inductor to decrease substrate losses.

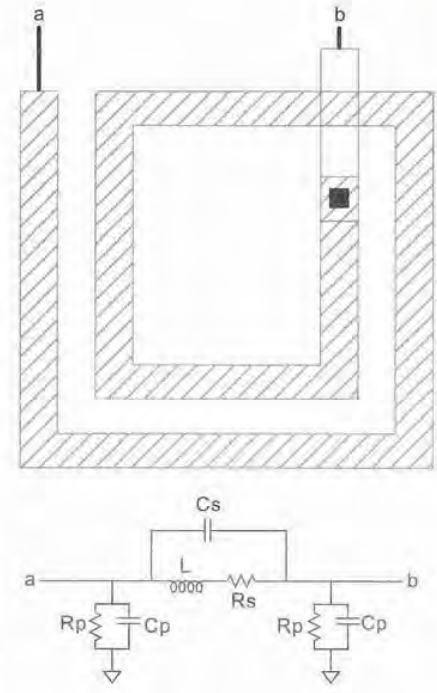


FIG 3.20 Typical spiral inductor and equivalent circuit [Rotella02]

3.4.3.4 Transmission Lines A *transmission line* can be used on a chip to provide a known impedance wire. Two basic kinds of transmission line are commonly used: *microstrip* and *coplanar waveguide*. These are shown in Figure 3.21.

A microstrip transmission line is shown in Figure 3.21(a). It is composed of a wire of width w placed over a ground plane and separated by a dielectric of height h and dielectric constant k. In the chip case, the wire might be the top level of metallization and the ground lane the next metal down.

A coplanar waveguide does not have to have a sublayer ground plane and is shown in Figure 3.21(b). It consists of a wire of width w spaced s on each side from *coplanar* ground wires. The reader is referred to [Wadell01] for detailed design equations.

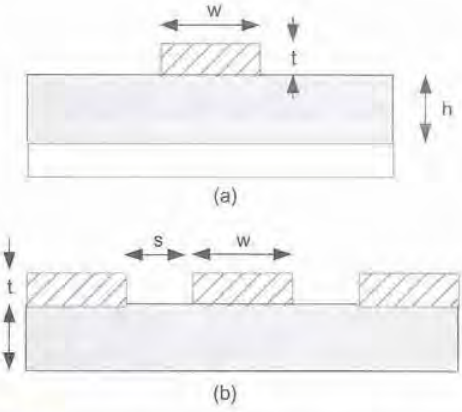


FIG 3.21 Microstrip and coplanar waveguide

3.4.3.5 Non-volatile Memory *Non-volatile* memory (NVM) retains its state when the power is removed from the circuit. The simplest NVM is a *mask-programmed* ROM cell (see Section 11.5.1). This type of NVM is not reprogrammable or programmable after the device is manufactured. A *one-time programmable* (OTP) memory can be implemented using a fuse constructed of a thin piece of metal through which is passed a current that vaporizes the metal by exceeding the current density in the wire. The first really reprogrammable memories used a stacked polysilicon gate structure and were programmed by applying a high voltage to the device in a manner that caused Fowler-Nordheim tunneling to store a charge on a floating gate. The whole memory could be

erased by exposing it to UV light that knocked the charge off the gate. These memories evolved to *electrically erasable* memories, which are today represented by *Flash* memory.

A typical Flash memory transistor is shown in Figure 3.22 [She02]. The source and drain structures can vary considerably to allow for high-voltage operation, but the dual-gate structure is fairly common. The gate structure is a stacked configuration commencing with a thin tunneling oxide. A floating polysilicon gate sits on top of this oxide and a conventional gate oxide is placed on top of the floating gate. Finally, a polysilicon control gate is placed on top of the gate oxide. The operation of the cell is also shown in Figure 3.22. In normal operation, the floating gate determines whether or not the transistor is conducting. To program the cell, the source is left floating and the control gate is raised to approximately 20 V (using an on-chip voltage multiplier). This causes electrons to tunnel into the floating gate, thus programming it. To deprogram a cell, the drain and source are left floating and the substrate (or well) is connected to 20 V. The electrons stored on the floating gate are attracted away, leaving the gate in an unprogrammed state.

3.4.3.6 Bipolar Transistors Bipolar transistors were mentioned previously in our discussion of SiGe process options. Both *npn* and *pnp* bipolar transistors can be added to a CMOS process, which is then called a BiCMOS process. These processes tend to be used

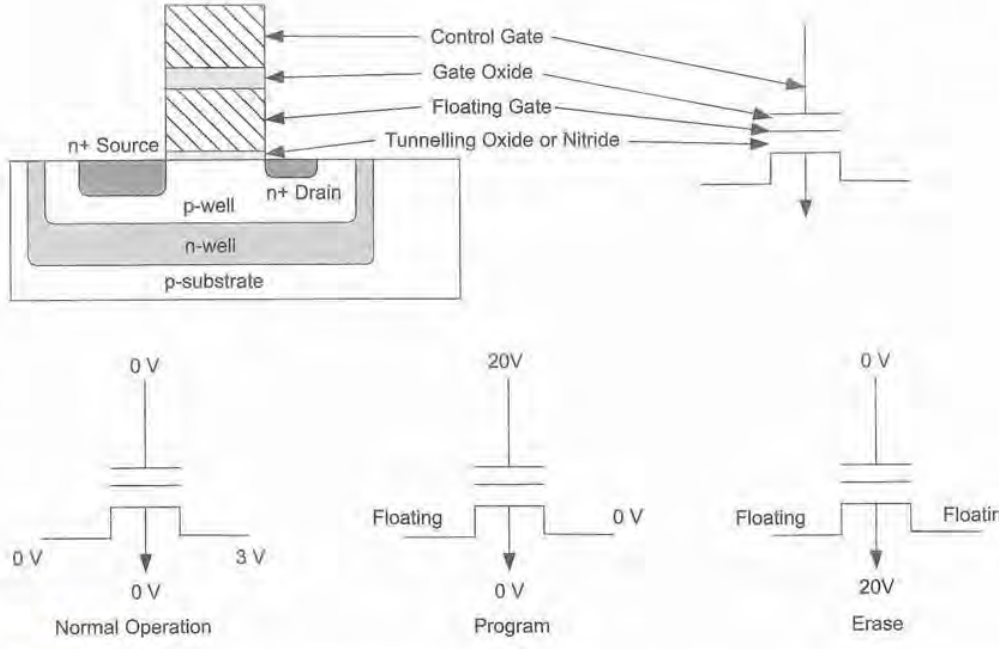


FIG 3.22 Flash memory construction and operation

for specialized analog or high-voltage circuits. In a regular n-well process, a parasitic pnp transistor is present that can be used for circuits such as bandgap voltage references. This transistor is shown in Figure 3.23 with the p-substrate collector, the n-well base, and the p-diffusion emitter. Both process cross-section and layout are shown. This transistor in conjunction with a parasitic npn is the cause of latchup (see Section 4.8.5).

3.4.3.7 Fuses and Antifuses The use of a fuse was mentioned in the section on nonvolatile memory. Fuses can be blown with a high current or zapped by a laser. In the latter case, an area is normally left in the passivation oxide to allow the laser direct access to the metal link that is to be cut. Figure 3.24 shows the layout of a metal fuse.

An *antifuse* is a device that initially has a high resistivity but can become low resistance when a programming voltage is applied. This device requires special processing and is used in programmable logic devices (see Section 8.3.2.2).

3.4.3.8 Micro Electro Mechanical Systems (MEMS) Semiconductor processes and especially CMOS processes have been used to construct tiny mechanical systems monolithically. A typical device is the well-known air-bag sensor, which is a small accelerometer

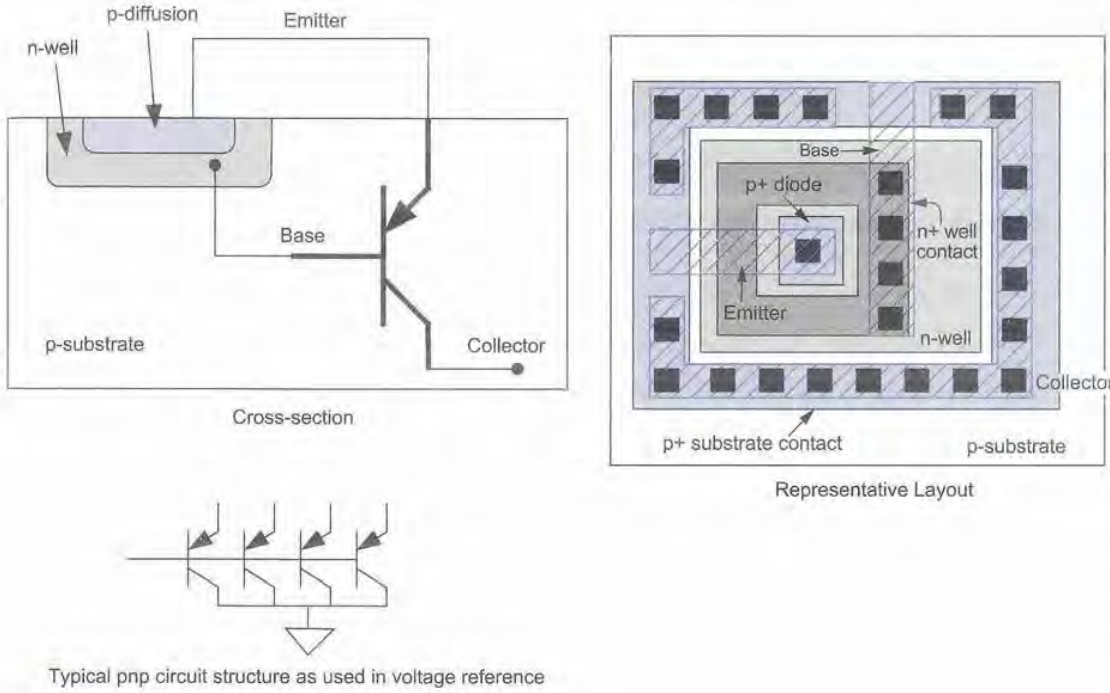


FIG 3.23 Parasitic pnp bipolar transistor

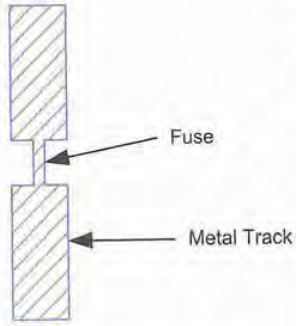


FIG 3.24 A typical metal fuse

consisting of an air bridge capacitor that can detect sudden changes in acceleration when co-integrated with some conditioning electronics. Structures such as cantilevers, mechanical resonators, and even micromotors have been built. A full discussion of MEMS is beyond the scope of this book, but further material can be found in [Maluf00, Kovacs98].

3.4.4 Beyond Conventional CMOS

Nanotechnology is presently a hot research area seeking alternative structures to replace CMOS when scaling finally runs out of steam. For example, carbon nanotubes have been used to demonstrate transistor behavior and build inverters [Liu01]. They are of interest as the nanotube is smaller than the predicted endpoint for CMOS gate lengths. A nanotube transistor is shown in Figure 3.25 [Collins01]. Presently, the speeds are quite slow and the manufacturing techniques are limited, but they may be of interest in the future.

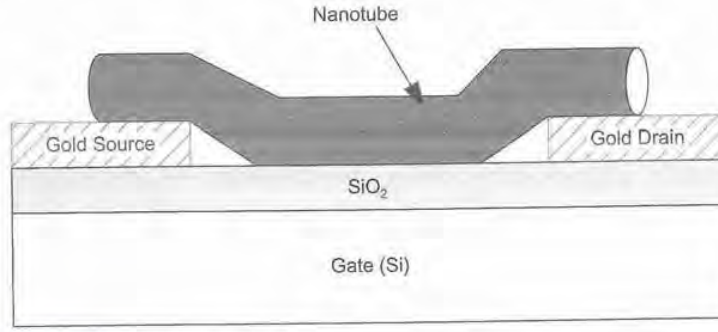


FIG 3.25 Carbon nanotube transistor

3.5 Technology-related CAD Issues

The mask database is the interface between the semiconductor manufacturer and the chip designer. Two basic checks have to be completed to ensure that this description can be turned into a working chip. First, the specified geometric design rules must be obeyed. Second, the interrelationship of the masks must, upon passing through the manufacturing process, produce the correct interconnected set of circuit elements. To check these two requirements, two basic CAD tools are required, namely a *Design Rule Check* (DRC) program and a mask circuit *extraction* program. The most common approach to implementing these tools is to provide a set of subprograms that perform general geometry operations. A particular set of DRC rules or extraction rules for a given CMOS process (or any semiconductor process) is then indicated by a specification of the operations that must be per-

formed on each mask and the inter-mask checks that must be completed. Accompanied by a written specification, these *run sets* are usually the defining specification for a process.

In this section we will examine a hypothetical DRC and extraction system to illustrate the nature of these run sets.

3.5.1 Design Rule Checking (DRC)

Although we can design the physical layout in a certain set of mask layers, the actual masks used in fabrication can be derived from the original specification. Similarly, when we want a program to determine what we have designed by examining the interrelationship of the various mask layers, it may be necessary to determine various logical combinations between masks.

To examine these concepts, let us posit the existence of the following functions (loosely based on the Cadence DRACULA DRC program), which we will apply to a geometric database (i.e., rectangles, polygons, paths):

AND `layer1 layer2 -> layer3`
ANDs layer1 and layer2 together to produce layer3
(i.e., the intersection of the two input mask descriptions).

OR `layer1 layer2 -> layer3`
ORs layer1 and layer2 together to produce layer3
(i.e., the union of the two input mask descriptions).

NOT `layer1 layer2 -> layer3`
Subtracts layer2 from layer1 to produce layer3
(i.e., the difference of the two input mask descriptions).

WIDTH `layer > dimension -> layer3`
Checks that all geometry on layer is larger than dimension.
Any geometry that is not is placed in layer3.

SPACE `layer > dimension -> layer3`
Checks that all geometry on layer is spaced further than dimension.
Any geometry that is not is placed in layer3.

The following layers will be assumed as input:

```
nwell
active
p-select
n-select
poly
poly-contact
active-contact
metal
```

Typically, useful sublayers are first generated. First, the four kinds of active area are isolated. The rule set to accomplish this is as follows:

```
NOT  all nwell -> substrate
AND  nwell active -> nwell-active
NOT  active nwell -> pwell-active
AND  nwell-active p-select -> pdiff
AND  nwell-active n-select -> vddn
AND  pwell-active n-select -> ndiff
AND  pwell-active p-select -> gndp
```

In the above specification, a number of new layers have been specified. For instance, the first rule states that wherever nwell is absent, a layer called substrate exists. The second rule states that all active areas within the nwell are nwell-active. A combination of nwell-active and p-select or n-select yields pdiff (p diffusion) or vddn (well tap).

To find the transistors, the following rule set is used:

```
AND poly ndiff -> ngates
AND poly pdiff -> pgates
```

The first rule states that the combination of polysilicon and ndiff yields the ngates region—all of the n-transistor gates.

Typical design rule checks (DRC) might include the following:

```
WIDTH metal  < 0.13 -> metal-width-error
SPACE metal  < 0.13 -> metal-space-error
```

For instance, the first rule determines if any metal is narrower than 0.13 μm and places the errors in the metal-width-error layer. This layer might be interactively displayed to highlight the errors.

3.5.2 Circuit Extraction

Now imagine that we want to determine the electrical connectivity of a mask database. The following commands are required:

CONNECT layer1 layer2
Electrically connect layer1 and layer2.

MOS name drain-layer gate-layer source-layer substrate-layer
Define an MOS transistor in terms of the component terminal layers. (This is, admittedly, a little bit of magic.)

The connections between layers can be specified as follows:

```
CONNECT active-contact pdiff
CONNECT active-contact ndiff
CONNECT active-contact vddn
CONNECT active-contact gndp
CONNECT active-contact metal
```

```
CONNECT gndp substrate
CONNECT vddn nwell
CONNECT poly-contact poly
CONNECT poly-contact metal
```

The connections between the diffusions and metal are specified by the first seven statements. The last two statements specify how metal is connected to poly.

Finally, the active devices are specified in terms of the layers that we have derived.

```
MOS nmos ndiff ngates ndiff substrate
MOS pmos pdiff pgates pdiff nwell
```

An output statement might then be used to output the extracted transistors in some netlist format (i.e., SPICE format). The extracted netlist is often used to compare the layout against the intended schematic.

It is important to realize that the above run set is manually generated. The data you extract from such a program is only as good as the input. For instance, if parasitic routing capacitances are required, then each layer interaction must be coded. If parasitic resistance is important in determining circuit performance, it also must be specifically included in the extraction run set.

3.6 Manufacturing Issues

As processes have evolved, various design rules have emerged that reflect the complexity of the processing. This section will cover some important areas.

3.6.1 Antenna Rules

When a metal wire contacted to a transistor gate is plasma-etched, it can charge up to a voltage sufficient to break down thin gate oxides. The metal can be contacted to diffusion to provide a path for the charge to bleed away. *Antenna rules* specify the maximum area of metal that can be connected to a gate without a source or drain to act as a discharge element. They are somewhat hard to visualize, but are fixed by placing jumpers to a higher layer of metal to shorten the metal segment or by placing diffusion diodes on wires. The design rule normally defines the maximum ratio of metal area to gate area such that charge on the metal will not damage the gate. The ratios can vary from 100:1 to 5000:1 depending on the thickness of the gate oxide (and hence breakdown voltage) of the transistor in question. Higher ratios apply to thicker gate oxide transistors (i.e., 3.3 V I/O transistors).

Figure 3.26 shows a typical fix to an antenna violation. In the top layout, the wire (L2) exceeds the antenna rule for the process. If this were to be connected to the gate, the gate would possibly break down in subsequent processing steps. The fix is shown in the bottom diagram, where a link to the top metal layer is made so that the length is now L1. At the point the source/drain connection is made, the gate is protected. This occurs at the final metallization step. The link has to be made at the level where a source/drain region

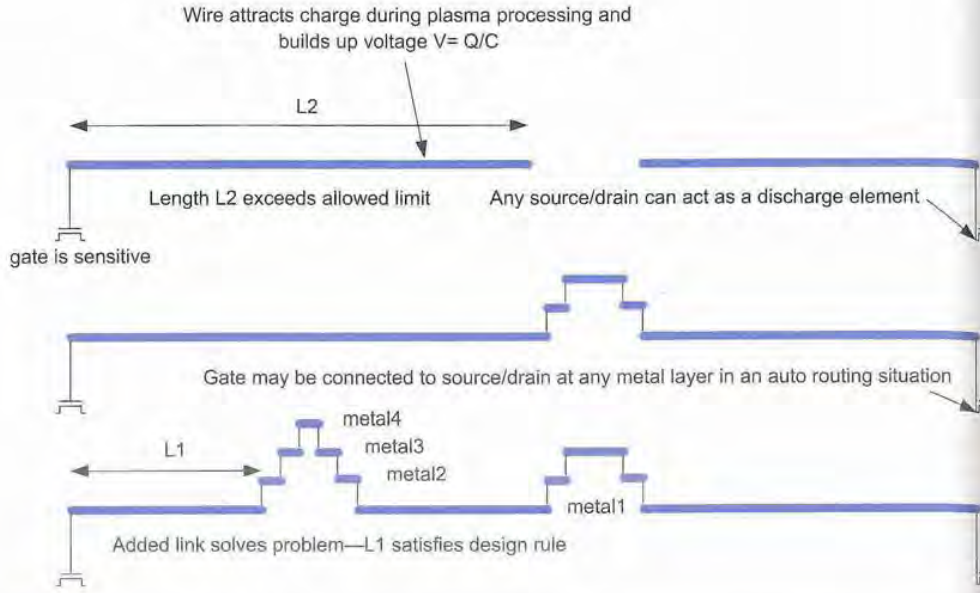


FIG 3.26 Antenna rule violation and fix

will connect to the gate. For safety's sake, in dense layouts it is normally assumed that the source/drain is connected at the final metallization step (in this case metal4). An alternative method is to attach source/drain diodes to problem nets as shown in Figure 3.27. These diodes can be simple junctions of n-diffusion to p-substrate rather than transistor source/drain regions.

3.6.2 Layer Density Rules

Another set of rules that pertain to advanced processes are layer density rules, which specify a minimum and maximum density of a particular layer within a specified area. These are required as a result of the CMP process and the desire to achieve uniform etch rates. For instance, a metal layer might have to have 30% minimum and 70% maximum fill within a 1 mm by 1 mm area. For digital circuits, these density levels are normally reached with normal routing. Analog and RF circuits, on the other hand, are almost by definition, sparse. Thus, gate and metal layers may have to be added manually or by a fill program after design has been completed. In some circumstances, the fill may have to be grounded via n-diffusion diodes. This is especially true for RF circuits where the metal1 fill can be used for a ground plane. Designers must be aware of the fill so that it does not introduce unexpected parasitic capacitance to nearby wires.

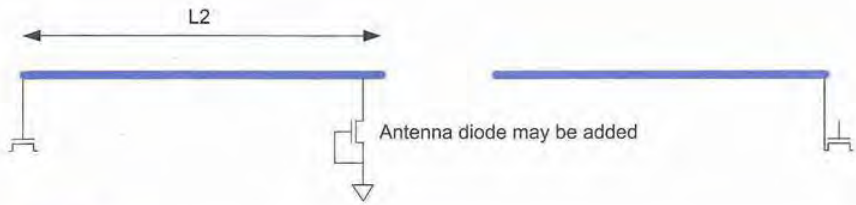


FIG 3.27 Antenna diode addition

3.6.3 Resolution Enhancement Rules

Some resolution enhancement techniques impose further design rules. For example, polysilicon typically uses the narrowest lines and thus needs the most enhancement. This can be simplest if polysilicon gates are only drawn in a single orientation (horizontal or vertical).

3.7 Pitfalls and Fallacies

Targeting a bleeding-edge process

There is a fine balance when you are deciding whether or not to move to a new process for a new design. On the one hand, you are tempted by increased density and speed. On the other hand, support for the new process can initially be expensive (becoming familiar with process rules, CAD tool scripts, porting analog and RF designs, locating logic libraries, etc.). In addition, CMOS foundries frequently tune their processes in the first few months of production, and often, yield improvement steps can reflect back to design rule changes that can impact designs late in their tapeout schedule. For this reason, it is frequently prudent not to jump immediately into a new process when it becomes available. On the other hand, if you are limited in speed or some other attribute that is solved by the new process, then you don't have much choice but to bite the bullet.

Using lambda design rules on commercial designs

Lambda rules have been used in this text for ease of explanation and consistency. They are useable for class designs. However, they are not very useful for production designs for deep sub-micron processes. Of particular concern are the metal width and spacing rules, which are too conservative for most production processes.

Failing to account for the parasitic effects of metal fill

With area density rules, particularly in metal, most design flows include an automatic fill step to achieve the correct metal density. Particularly in analog and RF circuits, it is important to either exclude the automatic fill operation from that area or check circuit performance

(continued)

after the fill by completing a full parasitic extract and rerunning the verification simulation scripts.

Failing to include process calibration test structures

It was mentioned in the discussion on scribe line structures that test structures are frequently inserted here by the silicon manufacturer. Documentation is often unavailable so it is prudent for designers (particularly in academic designs, which receive less support from a foundry) to include their own test structures such as transistors or ring oscillators. This allows designers to calibrate the silicon against simulation models.

3.8 Historical Perspective

In the last 30 years, we (especially those of us who are older in years) have lived through an amazing period of technology evolution in integrated circuits. While some hard limits are being approached in terms of lithography and device dimensions, history shows that the technology takes a path that leads to new devices, applications, and directions.

Summary

Being aware of CMOS process options and directions can greatly influence design decisions. Frequently, the combination of performance and cost possibilities in a new process can provide new product opportunities that were not available previously. Similarly, venerable processes can offer good opportunities with the right product.

One issue that has to be kept in mind is the ever-increasing cost of having a CMOS design fabricated in a leading-edge process. Mask cost for critical layers is in the vicinity of $100K per mask. A full mask set for an advanced process approaches $1M in cost. This in turn is reflected in the types of design and approaches to design that are employed for CMOS chips of the future. For instance, making a design programmable so that it can have a longer product life is a good first start. Chapter 8 covers these approaches in depth.

For more advanced reading on silicon processing, consult textbooks such as [Wolf00].

Exercises

3.1 A 248 nm UV step and scan machine costs $10M and can produce eighty 300 mm diameter, 90 nm node wafers per hour. A 157 nm UV step and scan machine costs $40M and can process twenty 300 mm diameter, 50 nm node wafers per hour. If the

machines have a depreciation period of four years, what is the difference in the cost per chip for a chip that occupies 50 square mm at 90 nm resolution if the stepper is used 10 times per process run for the critical layers?

3.2 If the gate oxide thickness in a SiO_2-based structure is 2 nm, what would be the thickness of a HfO_2-based dielectric providing the same capacitance?

3.3 Explain the difference between a polycide and a salicide CMOS process. Which would be likely to have higher performance and why?

3.4 Draw the layout for a pMOS transistor in an n-well process that has active, p-select, n-select, polysilicon, contact, and metal1 masks. Include the well contact to V_{DD}.

3.5 What is the best possible metal for interconnect? Why isn't it used?

3.6 Using Table 3.2, calculate the minimum contacted pitch as shown in Figure 3.28 for metal1, metal3, and metal6 in terms of lambda using the SUBM rules in a 6-layer process. Is there a wiring strategy that can reduce this pitch?

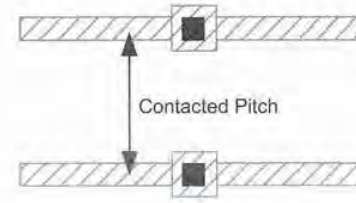


FIG 3.28 Contacted metal pitch

3.7 Using Table 3.2, calculate the minimum uncontacted and contacted transistor pitch as shown in Figure 3.29.

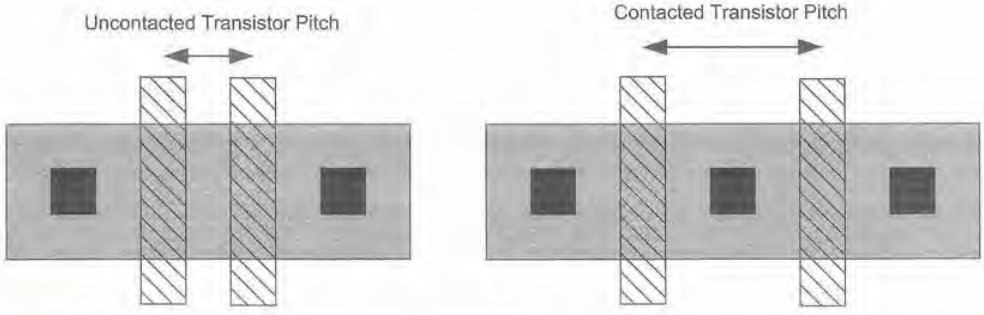


FIG 3.29 Uncontacted and contacted transistor pitch

3.8 Using Figure 3.30 and Table 3.2, calculate the minimum n to p pitch and the minimum inverter height with and without the poly contact to the gate (in). If an SOI process has 2 λ spacing between n and p diffusion, to what are the two pitches reduced?

3.9 Design a metal6 fuse ROM cell in a process where the minimum metal width is 0.5 μm and the maximum current density is 2 mA/μm. A fuse current of less than 10 mA is desired.

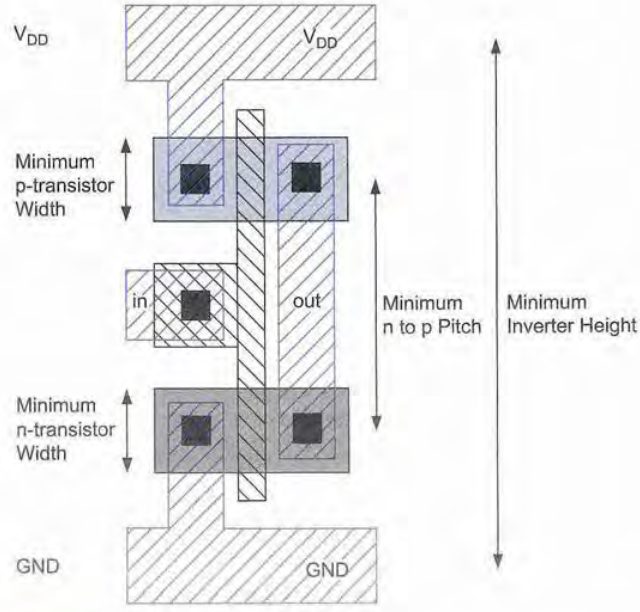


FIG 3.30 Minimum inverter height

4 Circuit Characterization and Performance Estimation

4.1 Introduction

In Chapter 1 we learned how to make chips that work. Now we move on to making chips that work *well*, where "well" can be defined as fast, low in power, inexpensive to manufacture and reliable. Before we can choose which design alternative is best, we must develop ways to estimate the goodness of each option, especially with regard to speed and power consumption.

The most obvious way to characterize a circuit is through simulation, and that will be the topic of Chapter 5. Unfortunately, simulations only inform us how a particular circuit behaves, not how to change the circuit to make it better. Moreover, if we don't know approximately what the result of the simulation should be, we are unlikely to catch bugs in our simulation model. Mediocre engineers rely predominantly on computer tools, but outstanding engineers develop their physical intuition to rapidly estimate the behavior of circuits.

In this chapter we are primarily concerned with the development of simple models that will assist us in the understanding of system performance. In a modern process, interconnect performance can be as important as or even more important than transistor performance. The issues to be considered in this chapter are

- Delay estimation in CMOS gates
- Power dissipation of CMOS logic
- Interconnect delay and signal integrity
- Design margining
- Reliability
- Effects of scaling

4.2 Delay Estimation

In most designs there will be many logic paths that do not require any conscious effort when it comes to speed. However, usually there will be a number of paths, called the *critical paths*, that require attention to timing details. These can be recognized by experience or timing simulation, but most designers use a timing analyzer, which is a design tool that automatically finds the slowest paths in a logic design (see Section 8.4.1.5). The critical paths can be affected at four main levels:

- The architectural/microarchitectural level
- The logic level
- The circuit level
- The layout level

The most leverage is achieved with a good microarchitecture. This requires a broad knowledge of both the algorithms that implement the function and the technology being targeted, such as how many gate delays fit in a clock cycle, how fast addition occurs, how fast memories are accessed, and how long signals take to propagate along a wire. Tradeoffs at the microarchitectural level include the number of pipeline stages, the number of execution units, and the size of memories.

The next level of timing optimization comes at the logic level. Tradeoffs include types of functional blocks (e.g., ripple carry vs. lookahead adders), the number of stages of gates in the cycle, and the fan-in and fan-out of the gates. The transformation from function to gates and registers can be done by experience, by experimentation, or increasingly by logic synthesis. Remember, however, that no amount of skillful logic design can overcome a poor microarchitecture.

Once the logic has been selected, the delay can be tuned at the circuit level by choosing transistor sizes or using other styles of CMOS logic. Finally, delay is dependent on the layout. The floorplan (either manually or automatically generated) is of great importance because it determines the wire lengths that can dominate delay. Tuning of particular cells can also reduce parasitic capacitance.

This section focuses on the logic and circuit optimizations of selecting the number of stages of logic, the types of gates, and the transistor sizes. Chapter 6 addresses other circuit styles and layout techniques. Chapters 10 and 11 examine design of datapath and array functional blocks.

Quick delay estimation is essential to designing critical paths. Although timing analyzers or circuit simulators can compute very detailed switching waveforms and accurately predict delay, good designers cannot be dependent on simulation alone. Simulation or timing analysis only answers how fast a particular circuit operates; they do not resolve the more interesting question of how the circuit could be modified to operate faster. Many novice designers spend countless hours tweaking parameters in a circuit simulator and resimulating only to find tiny improvements. Simple models that can be applied on the back of an envelope are important because they allow us to rapidly estimate delay,

understand its origin, and figure out how it can be reduced. This section applies the RC delay model to estimate the delay of logic gates.

We begin with a few definitions:

- *Rise time*, t_r = time for a waveform to rise from 20% to 80% of its steady-state value
- *Fall time*, t_f = time for a waveform to fall from 80% to 20% of its steady-state value
- *Edge rate*, $t_{rf} = (t_r + t_f)/2$
- *Propagation delay time*, t_{pd} = maximum time from the input crossing 50% to the output crossing 50%
- *Contamination delay time*, t_{cd} = minimum time from the input crossing 50% to the output crossing 50%

Intuitively, we know that when an input changes, the output will retain its old value for at least the contamination delay and take on its new value in at most the propagation delay. We sometimes differentiate between the delays for the output rising, t_{pdr}/t_{cdr} and the output falling, t_{pdf}/t_{cdf}. Rise/fall times are also sometimes called *slopes* or *edge rates*. Propagation and contamination delay times are also called *max-time* and *min-time*, respectively. The gate that charges or discharges a node is called the *driver* and the gates and wire being driven are called the *load*.

4.2.1 RC Delay Models

Section 2.6 developed a lumped RC model for transistors. Although transistors have complex nonlinear current-voltage characteristics, they can be approximated fairly well as a switch in series with a resistor, where the *effective resistance* is chosen to match the average amount of current delivered by the transistor. Transistor gate and diffusion nodes have capacitance.

In this section we apply the model to estimate the delay of logic gates as the RC product of the effective driver resistance and the load capacitance. Usually, logic gates use minimum-length devices for least delay, area, and power consumption. Given this, the delay of a logic gate depends on the widths of the transistors in the gate and the capacitance of the load that must be driven.

4.2.1.1 Effective Resistance and Capacitance Recall from Section 2.6 that an nMOS transistor with width of one unit is defined to have effective resistance R. The unit-width pMOS has a higher resistance that depends on its mobility relative to the nMOS transistor. For concreteness, let us assume this resistance is $2R$. Wider transistors have lower resistance. For example, a pMOS transistor of double-unit width has effective resistance R. Parallel and series transistors combine like conventional resistors. When multiple transistors are in series, their resistance is the sum of each individual resistance. When multiple transistors are in parallel, the resistance is lower if they are all ON. In many gates, the worst-case delay occurs when only one of several parallel transistors is ON. In that case, the effective resistance is just that of the single transistor.

Example

Sketch a 3-input NAND gate with transistor widths chosen to achieve effective rise and fall resistance equal to that of a unit inverter (R).

Solution: Figure 4.1 shows such a gate. The three nMOS transistors are in series so the resistance is three times that of a unit transistor. Therefore, each must be three times unit width to compensate. In other words, each transistor has resistance $R/3$ and the series combination has resistance R. The three pMOS transistors are in parallel. In the worst case (with one of the three inputs low), only one of the three pMOS transistors is ON. Therefore, each must be twice unit width to have resistance R.

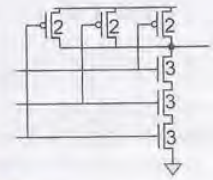


FIG 4.1 3-input NAND gate with unit rise and fall resistance

Also recall that the capacitance consists of gate capacitance and source/drain diffusion capacitance. Let us define the gate capacitance of a unit transistor to be C_g and the diffusion capacitance of its contacted source and drain to each be C_{diff}. In many processes the capacitances are approximately equal and can be labeled $C = C_g = C_{diff}$ to keep estimation simple. The second terminal of the diffusion capacitor is the body, which is usually tied to ground (for nMOS) or V_{DD} (for pMOS). As the DC voltage on the second terminal is irrelevant to delay, we often draw both capacitances to ground for simplicity. The gate capacitance includes fields terminating on the channel, source, and drain. To make hand analysis tractable, it can be approximated as a single capacitance to the V_{DD} or GND rail. C_g and C_{diff} are proportional to transistor width.

4.2.1.2 Diffusion Capacitance Layout Effects In a good layout, diffusion nodes are shared wherever possible to reduce the diffusion capacitance. Moreover, the uncontacted diffusion nodes between series transistors are usually smaller than those that must be contacted. Such uncontacted nodes have less capacitance (see Sections 2.3.3 and 5.4.4), although we will neglect the difference for hand calculations. —

Example

Annotate the 3-input NAND gate of Figure 4.1 with its gate and diffusion capacitances. Assume all diffusion nodes are contacted.

Solution: Figure 4.2(a) shows the gate and its capacitances. Each input drives five units of gate capacitance. Notice that the capacitors on source diffusions attached to the rails have both terminals shorted together so they are irrelevant to circuit operation. Figure 4.2(b) redraws the gate with these capacitances deleted and the remaining capacitances lumped to ground.

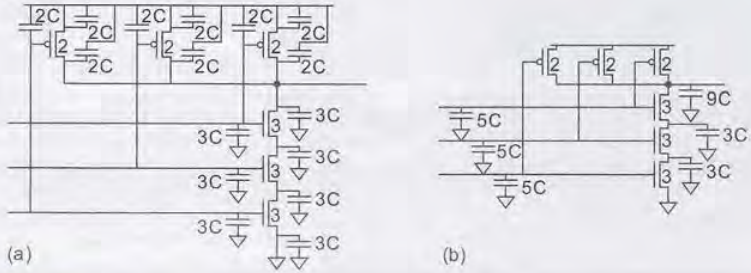


FIG 4.2 3-input NAND gate annotated with capacitances

In summary, gate capacitance can be determined directly from the transistor widths in the schematic. Diffusion capacitance depends on the layout. A conservative method of estimating capacitances before layout is to assume uncontacted diffusion between series transistors and contacted diffusion on all other nodes.

4.2.1.3 Elmore Delay Model Viewing ON transistors as resistors, we see that a chain of transistors can be represented as an RC ladder as shown in Figure 4.3. The Elmore delay [Elmore48] model estimates the delay of an RC ladder as the sum over each node in the ladder of the resistance R_{n-i} between that node and a supply multiplied by the capacitance on the node:

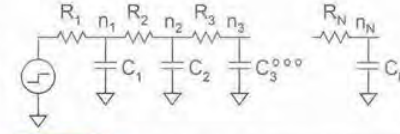


FIG 4.3 RC ladder for Elmore delay

$$t_{pd} = \sum_i R_{n-i} C_i = \sum_{i=1}^{N} C_i \sum_{j=i}^{i} R_j \qquad (4.1)$$

Example

Figure 4.4 shows a layout of the 3-input NAND gate. A single drain diffusion region is shared between two of the pMOS transistors. Estimate the actual diffusion capacitance from the layout.

Solution: Figure 4.5(a) shows the diffusion capacitance on the layout, neglecting those attached to the rails. Figure 4.5(b) redraws the schematic with these capacitances lumped to ground. Observe that two of the pMOS share a single diffusion region so the capacitance is smaller than predicted in Figure 4.2(b).

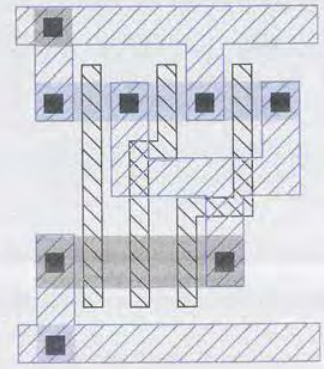


FIG 4.4 3-input NAND gate layout

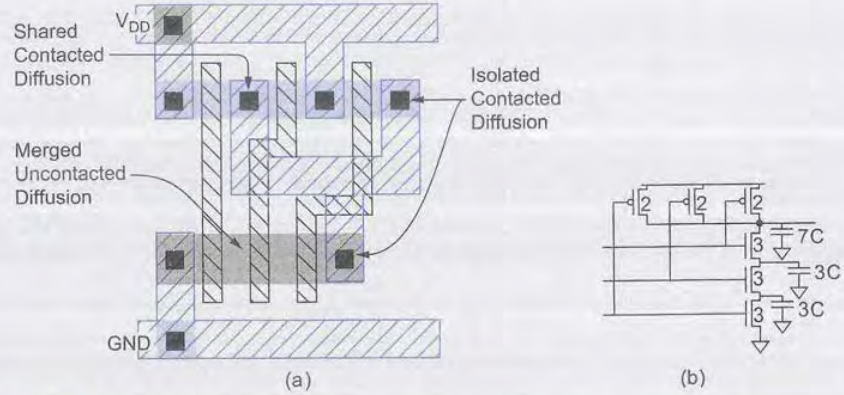


FIG 4.5 3-input NAND annotated with diffusion capacitances extracted from the layout

Example

Sketch a 2-input NAND gate with transistor widths chosen to achieve effective rise and fall resistance equal to a unit inverter. Compute the rising and falling propagation delays (in terms of R and C) of the NAND gate driving h identical NAND gates using the Elmore delay model. If C = 2 fF/μm and R = 2.5 kΩ • μm in a 180 nm process, what is the delay of a fanout-of-3 NAND gate?

Solution: Figure 4.6(a) shows such a NAND gate annotated with diffusion capacitance assuming contacted diffusion on each transistor except between the series nMOS transistors. The two nMOS transistors are in series so each must be made twice as wide to achieve overall unit resistance. The two pMOS transistors are in parallel. In the worst case, only one is ON, so it must be the same width as in the inverter, i.e., twice unit width because pMOS transistors have lower mobility than nMOS transistors. Each input is connected to 4 units of gate width. Hence the output load of h identical NAND gates may be represented as $4h$ units of capacitance. Figure 4.6(b) shows the equivalent circuit for estimating the rising delay. Only one pMOS transistor is ON in the slowest case. The diffusion capacitance of the nMOS transistor is ignored because it is not on the path between the supply rail and the output node Y. This results in a slightly optimistic result. The delay is $t_{pdr} = R \cdot ((6 + 4h)C) = (6 + 4h)RC$. Figure 4.6(c) shows the equivalent circuit for estimating the falling delay. In the worst case, input A is already '1,' so node x is charged up to nearly V_{DD} through the top nMOS transistor. Input B rises, turning on the bottom nMOS transistor and thus discharging both the capacitance on node x and the output capacitance. The Elmore delay is $t_{pdf} = (R/2)(2C) + R \cdot ((6 + 4h)C) = (7 + 4h)RC$. Despite the fact that the rise and fall resistances are equal, the falling propagation delay is slightly longer than the rising delay on account of the time needed to discharge the internal parasitic capacitance.

Observe that the best-case (contamination) delay of the gate could be substantially less. For example, if both inputs fell simultaneously, the output would be pulled up in half the time through the parallel combination of the two pMOS transistors: $t_{cdr} = (R/2)((6 + 4h)C) = (3 + 2h)RC$. If input B were already '1' and input A rises, node x would already have been discharged and thus could be ignored for delay purposes, reducing the falling delay to $t_{cdf} = R((6 + 4h)C) = (6 + 4h)RC$. Hence, for fastest response, the latest input should be connected to the transistor closest to the output node when feasible.

In the 180 nm process, RC = 5 ps. Therefore, a fanout-of-3 NAND gate (h = 3) has a delay of (6 + 4 • 3) • 5 ps = 90 ps.

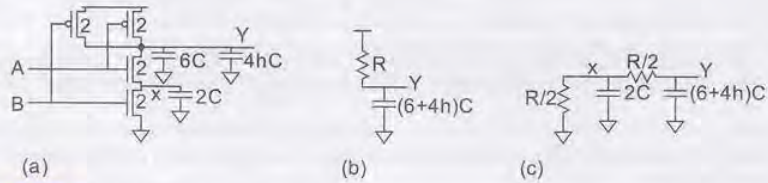


FIG 4.6 NAND gate delay estimation

Example

Suppose the widths of the transistors in the NAND gate are increased by a factor of k but the load is left unchanged at $4h'C$ (where $h' = h$ in the previous example). In other words, the load is equivalent to $h = h'/k$ NAND gates that are also a factor of k larger. Recompute the rising and falling propagation delays using the Elmore delay model.

Solution: Figure 4.7 shows the new equivalent circuits. The Elmore delay model predicts a rising delay of $(6 + 4(h'/k))RC = (6 + 4h)RC$ and a falling delay of $(7 + 4(h'/k))RC = (7 + 4h)RC$.

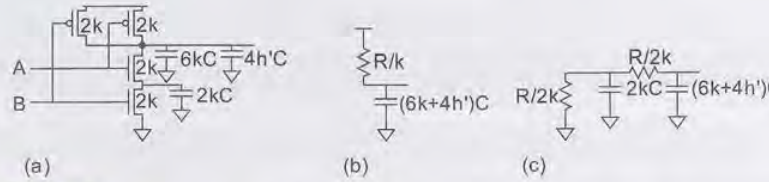


FIG 4.7 Scaled NAND gate delay estimation

Observe that the delay consists of two components. The *parasitic delay* of 6 or 7 is determined by the gate driving its own internal diffusion capacitance. Boosting the width of the transistors decreases the resistance but increases the capacitance so the parasitic delay is ideally independent of the gate size[1]. The *effort delay* of $4(h'/k)C = 4hC$ depends on the ratio (h) of external load capacitance to input capacitance and thus changes with transistor widths. The factor 4 is set by the complexity of the gate. The capacitance ratio is called the *electrical effort* or *fanout* and the term indicating gate complexity is called the *logical effort*. These components will be explored further in the subsequent sections.

It is often helpful to express delay in a process-independent form so that circuits can be compared based on topology rather than speed of the manufacturing process. Moreover, with a process-independent measure for delay, knowledge of circuit speeds gained while working in one process can be carried over to a new process. Recall that the delay of an ideal fanout-of-1 inverter with no parasitic capacitance is $\tau = 3RC$. We denote the normalized delay as multiples of this inverter delay: $d = t_{pd}/\tau$. Hence, the rising delay of the 2-input NAND gate is $d = (4/3)h + 2$. The RC delay model similarly predicts an inverter with real parasitics driving h identical inverters to have a delay of $h + 1$.

[1]Of course, gates with wider transistors may use layout tricks so the diffusion capacitance increases less than linearly with width, slightly decreasing the parasitic delay of large gates. Section 8.10.1.3 illustrates folding of wide transistors.

4.2.2 Linear Delay Model

In general the propagation delay of a gate can be written as

$$d = f + p \tag{4.2}$$

where p is the *parasitic delay* inherent to the gate when no load is attached; f is the *effort delay* or *stage effort* that depends on the complexity and fanout of the gate:

$$f = gh \tag{4.3}$$

The complexity is represented by the *logical effort*, g [Sutherland99]. An inverter is defined to have a logical effort of 1. More complex gates have greater logical efforts, indicating that they take longer to drive a given fanout. For example, the logical effort of the NAND gate from the previous example is 4/3. A gate driving h identical copies of itself is said to have a *fanout* or *electrical effort* of h. If the load is not identical copies of the gate, the electrical effort can be computed as

$$h = \frac{C_{out}}{C_{in}} \tag{4.4}$$

where C_{out} is the capacitance of the external load being driven and C_{in} is the input capacitance of the gate[2].

Figure 4.8 plots normalized delay vs. electrical effort for an idealized inverter and 2-input NAND gate. The y-intercepts indicate the parasitic delay, i.e., the delay when the gate drives no load. The slope of the lines is the logical effort. The inverter has a slope of 1 by definition. The NAND has a slope of 4/3.

The logical effort and parasitic delay can be estimated using RC models, as will be explored in the next sections, or extracted by curve-fitting simulated data, as discussed in Section 5.5.3. Logic gates fit the linear delay vs. fanout model remarkably well even in advanced processes; for example, Figure 5.28 shows agreement within 0.5 ps in a 180 nm process. A properly calibrated linear delay model is widely used by CAD tools such as logic synthesizers and static timing analyzers, although the notation varies from tool to tool. For example, the popular Synopsys Design Compiler tool uses the following basic model to define delay for a library of gates:

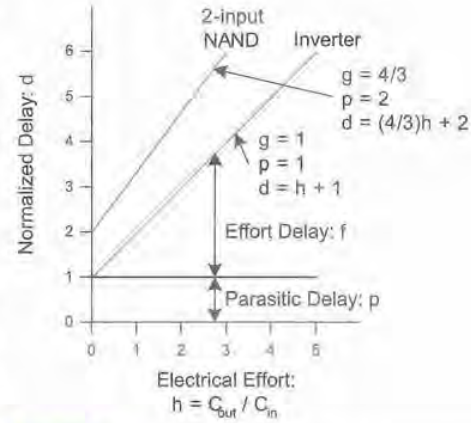


FIG 4.8 Normalized delay vs. fanout

```
delay_rise = intrinsic_rise + rise_resistance * capacitance
delay_fall = intrinsic_fall + fall_resistance * capacitance
```

[2]Some TTL designers say a gate has a fanout of h when it drives h other gates even if the other gates have different capacitances. This definition would not be useful for calculating delay and is best avoided in VLSI design. The term *electrical effort* avoids this potential confusion and emphasizes the parallels with logical effort.

Note that separate rising and falling delays are computed. These parameters are related to the logical effort terms as given in Table 4.1. The effective resistance of a gate increases with the logical effort of the gate but decreases with the gate size (i.e., input capacitance).

Table 4.1 Relationship between Logical Effort and Synopsys Terminology

Logical Effort Term	Synopsys Term
d	delay
p	intrinsic
C_{out}	capacitance
g/C_{in}	resistance

Some designers use the term *drive* as the reciprocal of resistance: drive = C_{in}/g. Gates with wider transistors have greater drive. Gates with high logical effort have less drive. Delay can be expressed in terms of drive as

$$d = \frac{C_{out}}{\text{drive}} + p \tag{4.5}$$

4.2.3 Logical Effort

Logical effort of a gate is defined as *the ratio of the input capacitance of the gate to the input capacitance of an inverter that can deliver the same output current.* Equivalently, logical effort indicates how much worse a gate is at producing output current as compared to an inverter, given that each input of the gate may only present as much input capacitance as the inverter.

Logical effort can be measured in simulation from delay vs. fanout plots as the ratio of the slope of the delay of the gate to the slope of the delay of an inverter. Alternatively, it can be estimated by sketching gates. Figure 4.9 shows inverter, NAND, and NOR gates with transistor widths chosen to achieve unit resistance, assuming pMOS transistors have twice the resistance of nMOS transistors[3]. The inverter presents 3 units of input capacitance. The NAND presents 4 units of capacitance on each input, so the logical effort is 4/3. Similarly, the NOR presents 5 units of capacitance, so the logical effort

[3]This assumption is made throughout the book. Exercises 4.19–4.20 explore the effects of different relative resistances (see also [Sutherland99]). The overall conclusions do not change very much, so the simple model is good enough for most hand estimates. A simulator or static timing analyzer should be used when more accurate results are required.

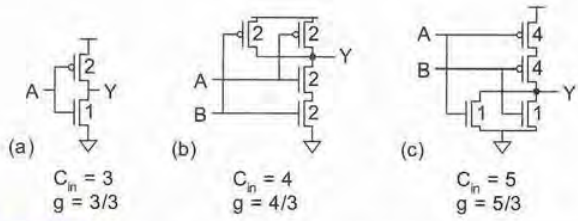


FIG 4.9 Logic gates sized for unit resistance

is 5/3. This matches our expectation that NANDs are better than NORs because NORs have slow pMOS transistors in series.

Table 4.2 lists the logical effort of common gates. The effort tends to increase with the number of inputs. NAND gates are better than NOR gates because the series transistors are nMOS rather than pMOS. Exclusive-OR gates are particularly costly and have different logical efforts for different inputs. An interesting case is that multiplexers built from ganged tristates, as shown in Figure 1.29(b), have a logical effort of 2 independent of the number of inputs. This might at first seem to imply that very large multiplexers are just as fast as small ones. However, the parasitic delay does increase with multiplexer size; hence, it is generally fastest to construct large multiplexers out of trees of 4-input multiplexers [Sutherland99].

Table 4.2 Logical effort of common gates

Gate Type	Number of Inputs				
	1	2	3	4	*n*
inverter	1				
NAND		4/3	5/3	6/3	$(n + 2)/3$
NOR		5/3	7/3	9/3	$(2n + 1)/3$
tristate, multiplexer	2	2	2	2	2
XOR, XNOR		4, 4	6, 12, 6	8, 16, 16, 8	

4.2.4 Parasitic Delay

The parasitic delay of a gate is the delay of the gate when it drives zero load. It can be estimated with RC delay models. A crude method good for hand calculations is to count only diffusion capacitance on the output node. For example, consider the gates in Figure 4.9, assuming each transistor on the output node has its own drain diffusion contact. Transis-

tor widths were chosen to give a resistance of R in each gate. The inverter has 3 units of diffusion capacitance on the output, so the parasitic delay is $3RC = \tau$. In other words, the normalized parasitic delay is 1. In general, we will call the normalized parasitic delay p_{inv}. p_{inv} is the ratio of diffusion capacitance to gate capacitance in a particular process. It is usually close to 1 and will be considered to be 1 on many examples for simplicity. The NAND and NOR each have 6 units of diffusion capacitance on the output, so the parasitic delay is twice as great ($2p_{inv}$, or simply 2). Table 4.3 estimates the parasitic delay of common gates. Increasing transistor sizes reduces resistance but increases capacitance correspondingly, so parasitic delay is, to first order, independent of gate size. However, wider transistors can be folded and often see less than linear increases in internal wiring parasitic capacitance, so in practice, larger gates tend to have slightly lower parasitic delay.

Table 4.3 Parasitic delay of common gates

Gate Type	Number of Inputs				
	1	2	3	4	n
inverter	1				
NAND		2	3	4	n
NOR		2	3	4	n
tristate, multiplexer	2	4	6	8	$2n$

This method of estimating parasitic delay is obviously crude. More refined estimates use the Elmore delay counting internal parasitics or extract the delays from simulation. The parasitic delay also depends on the ratio of diffusion capacitance to gate capacitance. For example, in a silicon-on-insulator process in which diffusion capacitance is much less, the parasitic delays will be lower. While knowing the parasitic delay is important for accurately estimating gate delay, we will see in Section 4.3 that the best transistor sizes for a particular circuit are only weakly dependent on parasitic delay. Hence, crude estimates may be acceptable.

Nevertheless, it is important to realize that parasitic delay grows more than linearly with the number of inputs in a real NAND or NOR circuit. For example, Figure 4.10 shows a model of an n-input NAND gate in which the upper inputs were all '1' and the bottom input rises. The gate must discharge the diffusion capacitances of all of the internal nodes as well as the output. The Elmore delay is

$$t_{pd} = R(3nC) + \sum_{i=1}^{n-1} \left(\frac{iR}{n}\right)(nC) = \left(\frac{n^2}{2} + \frac{5}{2}n\right)RC \tag{4.6}$$

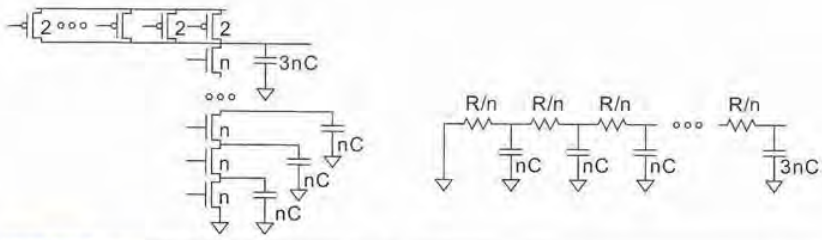


FIG 4.10 *n*-input NAND gate parasitic delay

This delay grows quadratically with the number of series transistors n, indicating that beyond a certain point it is faster to split a large gate into a cascade of two smaller gates. We will see in 4.2.5.3 that the coefficient of the n^2 term tends to be even larger in real circuits than in this simple model because of gate-source capacitance. In practice, it is rarely advisable to construct a gate with more than four or possibly five series transistors. When building large fan-in gates, trees of NAND gates are better than NOR gates because the nMOS transistors have lower resistance than pMOS transistors of the same size and capacitance.

4.2.5 Limitations to the Linear Delay Model



Logical effort is built on the linear delay model. Although the model works remarkably well for many practical applications, it also has limitations that should be understood when more accuracy is needed.

4.2.5.1 Input and Output Slope The largest source of error in the linear delay model is the input slope effect. Figure 4.11(a) shows a fanout-of-4 inverter driven by ramps with different slopes. Recall that the ON current increases with the gate voltage for an nMOS transistor. We say the transistor is *OFF* for $V_{gs} < V_t$, *fully ON* for $V_{gs} = V_{DD}$, and *partially ON* for intermediate gate voltages. As the rise time of the input increases, the delay also increases because the active transistor is not turned all the way ON at once. Figure 4.11(b) plots average inverter propagation delay vs. input rise time.

Notice that the delay vs. rise time data fits a straight line quite well. [Hedenstierna87] suggests that the line may be modeled as

$$t_{pd} = t_{pd-\text{step}} + t_{\text{edge}}\left(\frac{1+2\dfrac{|V_t|}{V_{DD}}}{6}\right) \tag{4.7}$$

where $t_{pd-\text{step}}$ is the propagation delay assuming a step input and t_{edge} is the appropriate edge rate (t_r or t_f).

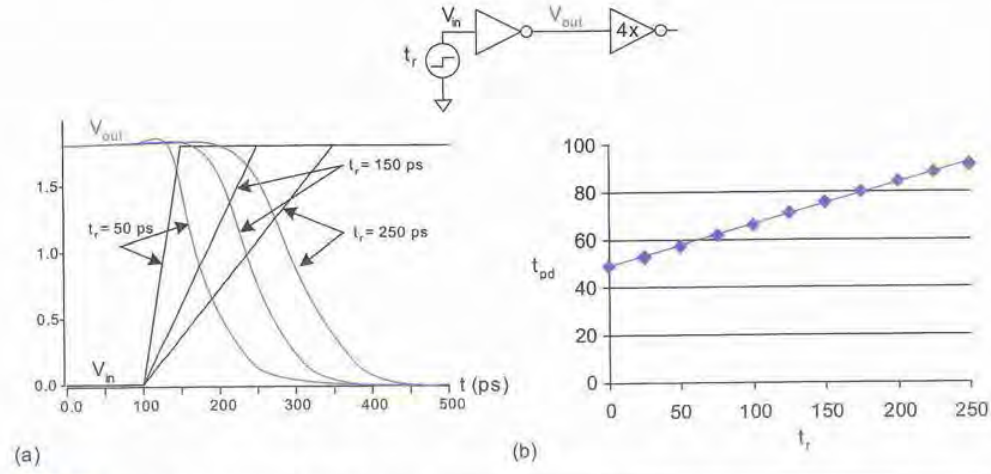


FIG 4.11 SPICE simulation of slope effect on CMOS inverter delay

A first-order RC circuit has a 20%-80% rise/fall time of $(\ln 0.8 - \ln 0.2)RC = 1.39RC$. The 20%–80% rise/fall time of a gate is roughly 1-1.5x the propagation delay. Hence, the input slope of a gate is related to the propagation delay of the previous gate. The Synopsys delay model supports considering the effect of slope on gate delay by introducing a slope_rise / slope_fall term. The overall delay of a gate becomes linearly dependent on the delay of the previous gate.

```
delay_rise = intrinsic_rise + rise_resistance · capacitance +
             slope_rise · delay_previous
delay_fall = intrinsic_fall + fall_resistance · capacitance +
             slope_fall · delay_previous
```

Accounting for slopes is important for accurate timing analysis, but is generally more complex than is worthwhile for hand calculations. Fortunately, we will see in Section 4.3 that circuits are fastest when each gate has the same effort delay and when that delay is roughly 4τ. Because slopes are related to edge rate, fast circuits tend to have relatively consistent slopes. If a cell library is characterized with these slopes, it will tend to be used in the regime in which it most accurately models delay.

4.2.5.2 Input Arrival Times Another source of error in the linear delay model is the assumption that one input of a multiple-input gate switches while the others are completely stable. When two inputs to a series stack turn ON simultaneously, the delay will be slightly longer than predicted because both transistors are only partially ON during the initial part of the transition. When two inputs to a parallel stack turn ON simulta-

neously, the delay will be shorter than predicted because both transistors deliver current to the output.

Figure 4.12 plots the propagation delay of an FO3 2-input NAND gate as a function of the input interarrival time. Input A switches at time 0, while input B switches in the same direction at time t_b. Propagation delay is measured from the latest input rising for falling outputs and from the earliest input falling for rising outputs. When one input arrives well before the other, $|t_b|$ is large and the propagation delay is essentially independent of t_b. This is the case assumed in the linear delay model. The delays are slightly different depending on which input arrives first, as will be explored in Section 5.5.3. When the two inputs arrive at nearly the same time, $|t_b|$ is small. t_{pdf} increases because of the series pulldowns while t_{pdr} decreases because of the parallel pullups.

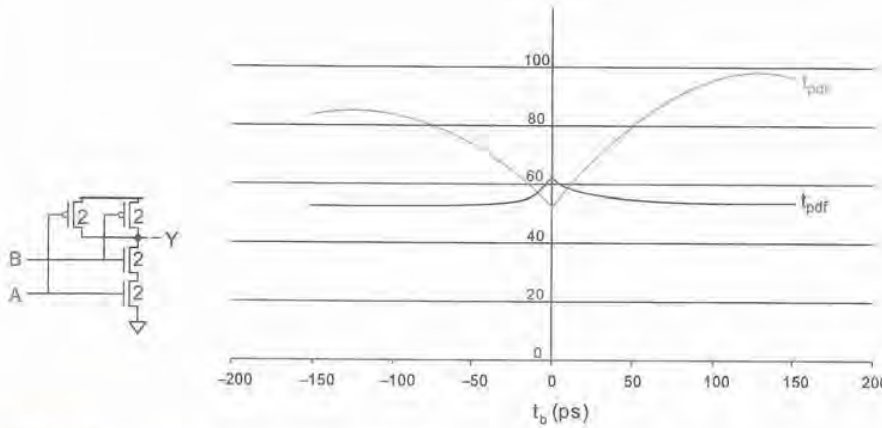


FIG 4.12 Delay sensitivity to input arrival time

4.2.5.3 Gate-Source Capacitance The examples in Section 4.2.1 assumed that gate capacitance terminates on a fixed supply rail. As discussed in Section 2.3.2, the bottom terminal of the gate oxide capacitor is the channel, which is primarily connected to the source when the transistor is ON. This means that as the source of a transistor changes value, charge is required to change the voltage on C_{gs}. Figure 4.13(a) revisits the 2-input NAND gate example, explicitly showing gate-source capacitances. As node x is discharged on a falling output transition in Figure 4.13(b), C_{gs2} must also be discharged. The delay can now be estimated as $(R/2)(2C + 2C) + R((2 + 2 + 2 + 4h)C) = (8 + 4h)RC$. RC models are more valuable for simplicity than accuracy, so some designers ignore this effect in hand calculations. Note that C_{gs} of the pMOS transistors does not affect delay because it is not on the path between Y and GND. However, the gate capacitance of both nMOS and pMOS would affect the loading and delay of the previous stage.

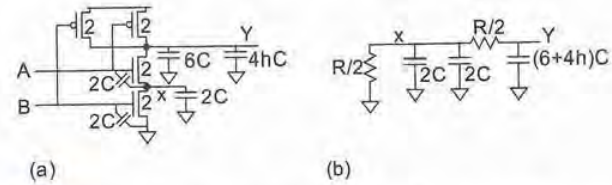


FIG 4.13 NAND gate delay estimation with gate-source capacitance modeled

4.2.5.4 Bootstrapping Transistors also have some capacitance from gate to drain. This capacitance couples the input and output in an effect known as *bootstrapping*, which can be understood by examining Figure 4.14(a). Our models so far have only considered C_{in} (C_{gs}). This figure also considers C_{gd}, the gate to drain capacitance. In the case that the input is rising (the output starts high), the effective input capacitance is $C_{gs} + C_{gd}$. When the output starts to fall, the voltage across C_{gd} changes, requiring the input to supply additional current to charge C_{gd}. In other words, the impact of C_{gd} on gate capacitance is effectively doubled.

To illustrate the effect of the bootstrap capacitance on a circuit, Figure 4.14(b) shows two inverter pairs. The top pair has an extra bit of capacitance between the input and output of the second inverter. The bottom pair has the same amount of extra capacitance from input to ground. When x falls, nodes a and c begin to rise (Figure 4.14(c)). At first, both nodes see approximately the same capacitance, consisting of the two transistors and the extra 3 fF. As node a rises, it initially bumps up b or "lifts b by its own bootstraps." Eventually the nMOS transistors turn ON, pulling down b and d. As b falls, it tugs on a

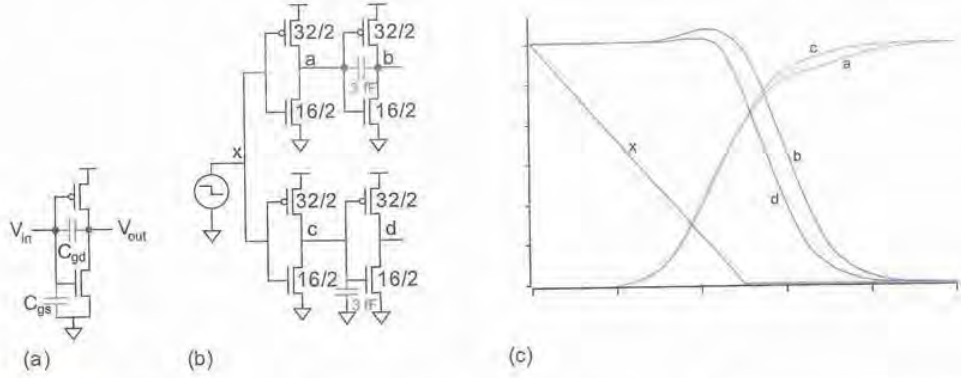


FIG 4.14 The effect of bootstrapping on inverter delay and waveform shape

through the capacitor, leading to the slow final transition visible on node *a*. Also observe that *b* falls later than *d* because of the extra charge that must be supplied to discharge the bootstrap capacitor. In summary, the extra capacitance has a greater effect when connected between input and output as compared to when it is connected between input and ground.

Because C_{gd} is fairly small, bootstrapping is only a mild annoyance in digital circuits. However, if the inverter is biased in its linear region near $V_{DD}/2$, the C_{gd} may appear multiplied by the gain of the inverter. This is known as the *Miller effect* and is of major importance in analog circuits.

4.3 Logical Effort and Transistor Sizing

Designers often need to choose the fastest circuit topology and gate sizes for a particular logic function and to estimate the delay of the design. As has been stated, simulation or timing analysis are poor tools for this task because they only determine how fast a particular implementation will operate, not whether the implementation can be modified for better results and if so, what to change. Inexperienced designers often end up in the "simulate and tweak" loop involving minor changes and many simulations. This is not only tedious but also seldom results in significant improvements. The method of Logical Effort [Sutherland99] provides a simple method "on the back of an envelope" to choose the best topology and number of stages of logic for a function. It allows the designer to quickly estimate the minimum possible delay for the given topology and to choose gate sizes that achieve this delay.

Logical Effort is based on the linear delay model. We first review using the model to estimate the delay of individual logic gates. The method generalizes to predict the delay of multistage logic networks and to choose the best number of stages for a multistage network. This section concludes with an example applying Logical Effort to design a memory decoder and summarizes the key insights from the method. The techniques of Logical Effort will be revisited throughout this text to understand delay of many types of circuits.

4.3.1 Delay in a Logic Gate

The linear delay model of EQs (4.2), (4.3), and (4.4) expresses propagation delay of a logic gate in terms of the complexity of a gate (its logical effort, *g*), the capacitive fanout (electrical effort, *h*), and the parasitic delay, *p*. Let us begin with two examples.

Example

Estimate the delay of the fanout-of-4 (FO4) inverter (i.e., an inverter driving four identical copies) shown in Figure 4.15. Assume the inverter is constructed in a 180 nm process with $\tau = 15$ ps.

Solution: The logical effort of the inverter is $g = 1$, by definition. The electrical effort is 4 because the load is four gates of equal size. The parasitic delay of an inverter is $p_{inv} \approx 1$. The total delay is $d = gh + p = 1 \cdot 4 + 1 = 5$ in normalized terms, or $t_{pd} = 75$ ps in absolute terms.

Often path delays are expressed in terms of FO4 inverter delays. While not all designers are familiar with the τ notation, most experienced designers do know the delay of a fanout-of-4 inverter in the process in which they are working. τ can be estimated as 0.2 FO4 inverter delays. Even if the ratio of diffusion capacitance to gate capacitance changes so $p_{inv} = 0.8$ or 1.2 rather than 1, the FO4 inverter delay only varies from 4.8 to 5.2. Hence, the delay of a gate-dominated logic block expressed in terms of FO4 inverters remains relatively constant from one process to another even if the diffusion capacitance does not.

As a rough rule of thumb, the FO4 delay for a process (in picoseconds) is 1/3 to 1/2 of the channel length (in nanometers). For example, a 180 nm process may have an FO4 delay of 60-90 ps. Delay is highly sensitive to process, voltage, and temperature variations, as will be examined in Section 6.6. The FO4 delay is usually quoted assuming typical process parameters and worst-case environment (low power supply voltage and high temperature).

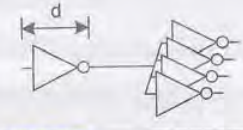


FIG 4.15 Fanout-of-4 (FO4) inverter

4.3.2 Delay in Multistage Logic Networks

Logical Effort generalizes to multistage logic networks. For example, Figure 4.16 shows the logical and electrical efforts of each stage in a multistage path as a function of the sizes of each stage. The path of interest (the only path in this case) is marked with the dashed blue line. Observe that logical effort is independent of size, while electrical effort depends on sizes.

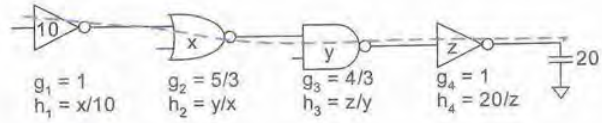


FIG 4.16 Multistage logic network

Example

A ring oscillator is constructed from an odd number of inverters, as shown in Figure 4.17. Estimate the frequency of an N-stage ring oscillator.

Solution: The logical effort of the inverter is $g = 1$, by definition. The electrical effort of each inverter is also 1 because it drives a single identical load. The parasitic delay is also 1. The delay of each stage is $d = gh + p = 1 \cdot 1 + 1 = 2$. An N-stage ring oscillator has a period of $2N$ stage delays because a value must propagate twice around the ring to regain the original polarity. Therefore, the period is $T = 2 \cdot 2N$. The frequency is the reciprocal of the period, $1/4N$.

A 31-stage ring oscillator in a 180 nm process has a frequency of $1/(4 \cdot 31 \cdot 15$ ps) = 540 MHz.

Note that ring oscillators are often used as process monitors to judge if a particular chip is faster or slower than nominally expected. One of the inverters should be replaced with a NAND gate to turn the ring off when not in use. The output can be routed to an external pad, possibly through a test multiplexer. The oscillation frequency should be low enough (e.g., 100 MHz) that the path to the outside world is not a limiter.

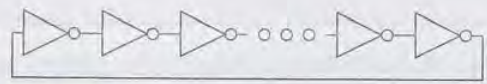


FIG 4.17 Ring oscillator

The *path logical effort* G can be expressed as the products of the logical efforts of each stage along the path.

$$G = \prod g_i \tag{4.8}$$

The *path electrical effort* H can be given as the ratio of the output capacitance the path must drive divided by the input capacitance presented by the path. This is more convenient than defining path electrical effort as the product of stage electrical efforts because we do not know the individual stage electrical efforts until gate sizes are selected.

$$H = \frac{C_{\text{out(path)}}}{C_{\text{in(path)}}} \tag{4.9}$$

The *path effort F* is the product of the stage efforts of each stage. Can we by analogy state $F = GH$?

$$F = \prod f_i = \prod g_i h_i \tag{4.10}$$

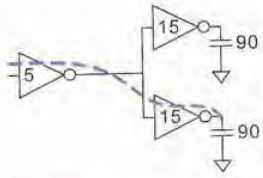


FIG 4.18 Circuit with two-way branch

In paths that branch, $F \neq GH$. This is illustrated in Figure 4.18, a circuit with a two-way branch. Consider a path from the primary input to one of the outputs. The path logical effort is $G = 1 \cdot 1 = 1$. The path electrical effort is $H = 90/5 = 18$. Thus, $GH = 18$. But $F = f_1 f_2 = g_1 h_1 g_2 h_2 = 1 \cdot 6 \cdot 1 \cdot 6 = 36$. In other words, $F = 2GH$ in this path on account of the two-way branch.

We must introduce a new kind of effort to account for branching between stages of a path. This *branching effort b* is the ratio of the total capacitance seen by a stage to the capacitance on the path; in Figure 4.18 it is (15+15)/15 = 2.

$$b = \frac{C_{\text{onpath}} + C_{\text{offpath}}}{C_{\text{onpath}}} \tag{4.11}$$

The *path branching effort B* is the product of the branching efforts between stages.

$$B = \prod b_i \tag{4.12}$$

Now we can define the path effort F as the product of the logical, electrical, and branching efforts of the path. Note that the product of the electrical efforts of the stages is actually BH, not just H.

$$F = GBH \tag{4.13}$$

We can now compute the delay of a multistage network. The *path delay D* is the sum of the delays of each stage. It can also be written as the sum of the *path effort delay* D_F and *path parasitic delay P*.

$$\begin{aligned} D &= \sum d_i = D_F + P \\ D_F &= \sum f_i \\ P &= \sum p_i \end{aligned} \tag{4.14}$$

The product of the stage efforts is F, independent of gate sizes. The path effort delay is the sum of the stage efforts. The sum of a set of numbers whose product is constant is

Example

Estimate the minimum delay of the path from A to B in Figure 4.19 and choose transistor sizes to achieve this delay. The initial NAND2 gate may present a load of 8 λ of transistor width on the input and the output load is equivalent to 45 λ of transistor width.

Solution: The path logical effort is $G = (4/3) \cdot (5/3) \cdot (5/3) = 100/27$. The path electrical effort is $H = 45/8$. The path branching effort is $B = 3 \cdot 2 = 6$. The path effort is $F = GBH = 125$. As there are three stages, the best stage effort is $\hat{f} = \sqrt[3]{125} = 5$. The path parasitic delay is $P = 2 + 3 + 2 = 7$. Hence, the minimum path delay is $D = 3 \cdot 5 + 7 = 22$ in units of τ, or 4.4 FO4 inverter delays. The gate sizes are computed with the capacitance transformation: $y = 45 \cdot (5/3)/5 = 15$. $x = (15 + 15) \cdot (5/3)/5 = 10$. We check that the initial 2-input NAND gate should have a size of $(10 + 10 + 10) \cdot (4/3)/5 = 8$, as desired. The transistor sizes in Figure 4.20 are chosen to give the desired amount of input capacitance while achieving equal rise and fall delays.

We can also check that our delay was achieved. The NAND2 gate delay is $d_1 = g_1 h_1 + p_1 = (4/3) \cdot (10 + 10 + 10)/8 + 2 = 7$. The NAND3 gate delay is $d_2 = g_2 h_2 + p_2 = (5/3) \cdot (15 + 15)/10 + 3 = 8$. The NOR2 gate delay is $d_3 = g_3 h_3 + p_3 = (5/3) \cdot 45/15 + 2 = 7$. Hence, the path delay is 22, as predicted.

Many inexperienced designers know that wider transistors offer more current and thus try to make circuits faster by using bigger gates. Increasing the size of any of the gates except the first one only makes the circuit slower. For example, increasing the size of the NAND3 makes the NAND3 faster but makes the NAND2 slower, resulting in a net speed loss. Increasing the size of the initial NAND2 gate does speed up the circuit under consideration. However, it presents a larger load on the path that computes input A, making that path slower. Hence, it is crucial to have a specification of not only the load the path must drive but also the maximum input capacitance the path may present.

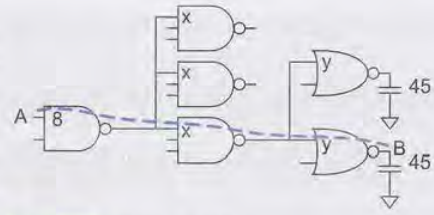


FIG 4.19 Example path

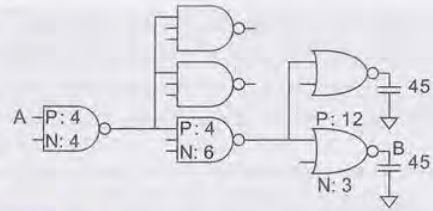


FIG 4.20 Example path annotated with transistor sizes

minimized by choosing all the numbers to be equal. In other words, the path delay is minimized when each stage bears the same effort $\hat{f}$. If a path has N stages and each bears the same effort, that effort must be

$$\hat{f} = g_i h_i = F^{1/N} \tag{4.15}$$

Thus the minimum possible delay of an N-stage path with path effort F and path parasitic delay P is

$$D = NF^{1/N} + P \tag{4.16}$$

This is a key result of Logical Effort. It shows that the minimum delay of the path can be estimated knowing only the number of stages, path effort, and parasitic delays without the need to assign transistor sizes. This is superior to simulation, in which delay depends on sizes and you never achieve certainty that the sizes selected are those that offer minimum delay.

It is also straightforward to select gate sizes to achieve this least delay. Combining EQs (4.3) and (4.4) gives us the *capacitance transformation* formula to find the best input capacitance for a gate given the output capacitance it drives.

$$C_{\text{in}_i} = \frac{C_{\text{out}_i} * g_i}{\hat{f}} \tag{4.17}$$

Starting with the load at the end of the path, work backward applying the capacitance transformation to determine the size of each stage. Check the arithmetic by verifying that the size of the initial stage matches the specification.

4.3.3 Choosing the Best Number of Stages

Given a specific circuit topology, we now know how to estimate delay and choose gate sizes. However, there are many different topologies that implement a particular logic function. Logical Effort tells us that NANDs are better than NORs and that gates with few inputs are better than gates with many. In this section we will also use Logical Effort to predict the best number of stages to use.

Logic designers sometimes estimate delay by counting the number of stages of logic, assuming each stage has a constant "gate delay." This is potentially misleading because it implies that the fastest circuits are those that use the fewest stages of logic. Of course the gate delay actually depends on the electrical effort, so sometimes using fewer stages results in more delay. The following example illustrates this point.

In general, you can always add inverters to the end of a path without changing its function (save possibly for polarity). Let us compute how many should be added for least delay. The logic block shown in Figure 4.22 has n_1 stages and a path effort of F. Consider adding $N - n_1$ inverters to the end to bring the path to N stages. The extra inverters do not

Example

A control unit generates a signal from a unit-sized inverter. The signal must drive unit-sized loads in each bitslice of a 64-bit datapath. The designer can add inverters to buffer the signal to drive the large load. Assuming polarity of the signal does not matter, what is the best number of inverters to add and what delay can be achieved?

Solution: Figure 4.21 shows the cases of adding 0, 1, 2, or 3 inverters. The path electrical effort is $H = 64$. The path logical effort is $G = 1$, independent of the number of inverters. Thus the path effort is $F = 64$. The inverter sizes are chosen to achieve equal stage effort. The total delay is $D = N\sqrt[N]{64} + N$.

The 3-stage design is fastest and is much superior to a single stage. If an even number of inversions were required, the two- or four-stage designs are promising. The four-stage design is slightly faster, but the two-stage design requires significantly less area and power.

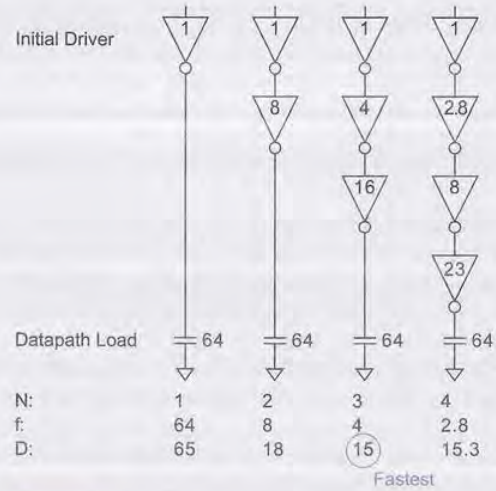


FIG 4.21 Comparison of different number of stages of buffers

change the path logical effort but do add parasitic delay. The delay of the new path is

$$D = NF^{1/N} + \sum_{i=1}^{n_1} p_i + (N - n_1)p_{\text{inv}} \qquad (4.18)$$

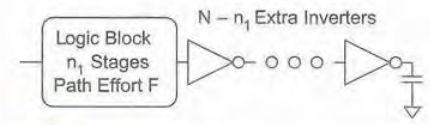


FIG 4.22 Logic block with additional inverters

Differentiating with respect to N and setting to 0 allows us to solve for the best number of stages. The result can be expressed more compactly by defining

$$\rho = F^{1/\hat{N}}$$

to be the best stage effort.

$$\frac{\partial D}{\partial N} = -F^{1/N} \ln F^{1/N} + F^{1/N} + p_{inv} = 0 \tag{4.19}$$
$$\Rightarrow p_{inv} + \rho(1 - \ln \rho) = 0$$

EQ (4.19) has no closed form solution. Neglecting parasitics (i.e., assuming $p_{inv} = 0$), we find the classic result that $\rho = 2.71828$ (e) [Mead80]. In practice, the parasitic delays mean each inverter is somewhat more costly to add. As a result, it is better to use fewer stages, or equivalently a higher stage effort than e. Solving numerically, when $p_{inv} = 1$, we find $\rho = 3.59$.

A path achieves least delay by using $\hat{N} = \log_\rho F$ stages. It is important to understand not only the best stage effort and number of stages but also the sensitivity to using a different number of stages. Figure 4.23 plots the delay increase using a particular number of stages against the number of stages, for $p_{inv} = 1$. The curve is very flat around the optimum. The delay is within 15% of the best achievable if the number of stages is within 2/3 to 1.5 times the theoretical best number (i.e., ρ is in the range of 2.4 to 6).

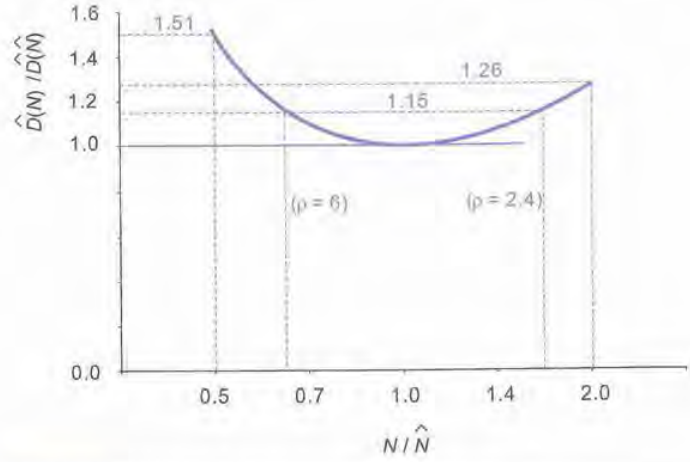


FIG 4.23 Sensitivity of delay to number of stages

Using a stage effort of 4 is a convenient choice and simplifies mentally choosing the best number of stages. This effort gives delays within 2% of minimum for p_{inv} in the range of 0.7 to 2.5. This further explains why a fanout-of-4 inverter has a "representative" logic gate delay.

4.3.4 Example

Consider a larger example to illustrate the application of Logical Effort. Our esteemed colleague Ben Bitdiddle is designing a decoder for a register file in the Motoroil 68W86, an embedded processor for automotive applications. The decoder has the following specifications:

- 16-word register file
- 32-bit words
- Each register bit presents a load of 3 unit-sized transistors on the word line (2 unit-sized access transistors plus some wire capacitance)
- True and complementary versions of the address bits $A[3:0]$ are available
- Each address input can drive 10 unit-sized transistors

As we will see further in Section 11.2.2, a 2^N-word decoder consists of 2^N N-input AND gates. Therefore, the problem is reduced to designing a suitable 4-input AND gate. Let us help Ben determine how many stages to use, how large each gate should be, and how fast the decoder can operate.

The output load on a word line is 32 bits with 3 units of capacitance each, or 96 units. Therefore, the path electrical effort is $H = 96/10 = 9.6$. Each address is used to compute half of the 16 word lines; its complement is used for the other half. Therefore, a $B = 8$-way branch is required somewhere in the path. Now we are faced with a chicken-and-egg dilemma. We need to know the path logical effort to calculate the path effort and best number of stages. However, without knowing the best number of stages, we cannot sketch a path and determine the logical effort for that path. There are two ways to resolve the dilemma. One is to sketch a path with a random number of stages, determine the path logical effort, then use that to compute the path effort and the actual number of stages. The path can be redesigned with this number of stages, refining the path logical effort. If the logical effort changes significantly, the process can be repeated. Alternatively, we know that the logic of a decoder is rather simple so we can ignore the logical effort (assume $G = 1$). Then we can proceed with our design, remembering that the best number of stages is likely slightly higher than predicted because we neglected logical effort.

Taking the second approach, we find the path effort is $F = GBH = (1)(8)(9.6) = 76.8$. Targeting a best stage effort of $\rho = 4$, we find the best number of stages is $N = \log_4 76.8 = 3.1$. Let us select a 3-stage design, recalling that a 4-stage design might be a good choice too when logical effort is considered. Figure 4.24 shows a possible 3-stage design (INV-NAND4-INV).

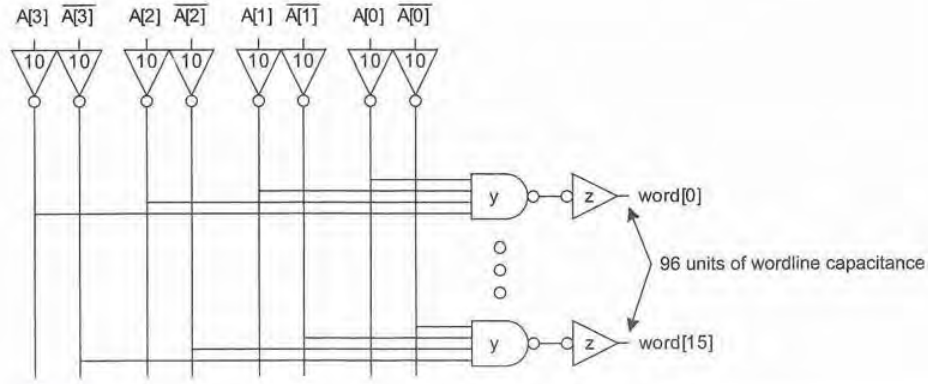


FIG 4.24 3-stage decoder design

The path has a logical effort of $G = 1 \cdot (6/3) \cdot 1 = 2$, so the actual path effort is $F = (2)(8)(9.6) = 154$. The stage effort is $\hat{f} = 154^{1/3} = 5.36$. This is in the reasonable range of 2.4 to 6, so we expect our design to be acceptable. Applying the capacitance transformation, we find gate sizes $z = 96 \cdot 1/5.36 = 18$ and $y = 18 \cdot 2/5.36 = 6.7$. The delay is $3 \cdot 5.36 + 1 + 4 + 1 = 22.1$.

Logical Effort also allows us to rapidly compare alternative designs using a spreadsheet rather than a schematic editor and a large number of simulations. Table 4.4 compares a number of alternative designs. We find a 4-stage design is somewhat faster, as we suspected. The 4-stage NAND2-INV-NAND2-INV design not only has the theoretical best number of stages but also uses simpler 2-input gates to reduce the logical effort and parasitic delay to obtain a 12% speedup over the original design. However, the 3-stage design has a smaller total gate area and dissipates less power.

Table 4.4 Spreadsheet comparing decoder designs

Design	Stages *N*	*G*	*P*	*D*
NAND4-INV	2	2	5	29.8
NAND2-NOR2	2	20/9	4	30.1
INV-NAND4-INV	**3**	**2**	**6**	**22.1**
NAND4-INV-INV-INV	4	2	7	21.1
NAND2-NOR2-INV-INV	4	20/9	6	20.5
NAND2-INV-NAND2-INV	4	16/9	6	19.7
INV-NAND2-INV-NAND2-INV	5	16/9	7	20.4
NAND2-INV-NAND2-INV-INV-INV	6	16/9	8	21.6

4.3.5 Summary and Observations

Logical Effort provides an easy way to compare and select circuit topologies, choose the best number of stages for a path, and estimate path delay. The notation takes some time to become natural but this author has poured through all the letters in the English and Greek alphabets without finding better notation. It may help to remember d for "**d**elay," p for "parasitic," b for "**b**ranching," f for "**e**ffort," g for "lo**g**ical effort," (or perhaps **g**ain), and h as the next letter after "f" and "g." The notation is summarized in Table 4.5 for both stages and paths.

Table 4.5 Summary of Logical Effort notation

Term	Stage Expression	Path Expression
number of stages	1	N
logical effort	g (see Table 4.2)	$G = \prod g_i$
electrical effort	$h = \frac{C_{out}}{C_{in}}$	$H = \frac{C_{out(path)}}{C_{in(path)}}$
branching effort	$b = \frac{C_{onpath} + C_{offpath}}{C_{onpath}}$	$B = \prod b_i$
effort	$f = gh$	$F = GBH$
effort delay	f	$D_F = \sum f_i$
parasitic delay	p (see Table 4.3)	$P = \sum p_i$
delay	$d = f + p$	$D = \sum d_i = D_F + P$

The method of Logical Effort is applied with the following steps:

1. Compute the path effort: $F = GBH$
2. Estimate the best number of stages: $\hat{N} = \log_4 F$

3. Sketch a path using: $\hat{N}$ stages
4. Estimate the minimum delay: $D = \hat{N}F^{1/\hat{N}} + P$
5. Determine the best stage effort: $\hat{f} = F^{1/\hat{N}}$
6. Starting at the end, work backward to find sizes: $C_{\text{in}_i} = \dfrac{C_{\text{out}_i} \cdot g_i}{\hat{f}}$

CAD tools are very fast and accurate at evaluating complex delay models, so Logical Effort should not be used as a replacement for such tools. Rather, its value arises from "quick and dirty" hand calculations and from the insights it lends to circuit design. Some of the key insights include:

- The idea of a numeric "logical effort" that characterizes the complexity of a logic gate or path allows you to compare alternative circuit topologies and show that some topologies are better than others.
- NAND structures are faster than NOR structures in complementary CMOS circuits.
- Paths are fastest when the effort delays of each stage are about the same and when these delays are close to four.
- Path delay is insensitive to modest deviations from the optimum. Stage efforts of 2.4–6 give designs within 15% of minimum delay. There is no need to make calculations to more than 1–2 significant figures, so many estimations can be made in your head. There is no need to choose transistor sizes exactly according to theory and there is little benefit in tweaking transistor sizes if the design is reasonable.
- Using stage efforts somewhat greater than 4 reduces area and power consumption at a slight cost in speed. Using efforts greater than 6-8 comes at a significant cost in speed.
- Using fewer stages for "less gate delays" does not make a circuit faster. Making gates larger also does not make a circuit faster; it only increases the area and power consumption.
- The delay of a well-designed path is about $\log_4 F$ fanout-of-4 (FO4) inverter delays. Each quadrupling of the load adds about one FO4 inverter delay to the path. Control signals fanning out to a 64-bit datapath therefore incur an amplification delay of about 3 FO4 inverters.
- The logical effort of each input of a gate increases through no fault of its own as the number of inputs grows. Considering both logical effort and parasitic delay, we find a practical limit of about 4 series transistors in logic gates and about 4 inputs to multiplexers. Beyond this fan-in, it is faster to split gates into multiple stages of skinnier gates.

- Inverters or 2-input NAND gates with low logical efforts are best for driving nodes with a large branching effort. Use small gates after the branches to minimize load on the driving gate.
- When a path forks and one leg is more critical than the others, buffer the noncritical legs to reduce the branching effort on the critical path.

4.3.6 Limitations of Logical Effort



Logical Effort is based on the linear delay model and the simple premise that making the effort delays of each stage equal minimizes path delay. This simplicity is the method's greatest strength but also results in a number of limitations:

- The linear delay model fails to capture the effect of input slope. Fortunately, edge rates tend to be about equal in well-designed circuits with equal effort delay per stage.
- The RC delay model neglects the effects of velocity saturation and overestimates the logical effort of NAND structures. It also ignores the body effect. Logical effort may be more accurately characterized through simulation, as shown in Section 5.5.3.
- Logical Effort does not account for interconnect. The effects of nonnegligible wire capacitance and RC delay will be revisited in Section 4.5. Logical Effort is most applicable to high-speed circuits with regular layouts where routing delay does not dominate. Such structures include adders, multipliers, memories, and other datapaths and arrays.
- Logical Effort explains how to design a path for maximum speed but not how to design for minimum area or power given a fixed speed constraint.
- Paths with complex branching are difficult to analyze by hand.

4.3.7 Extracting Logical Effort from Datasheets



When using a standard cell library, you can often extract logical effort of gates directly from the datasheets. For example, Figure 4.25 shows the INV and NAND2 datasheets from the Artisan Components library for the TSMC 180 nm process. The gates in the library come in various drive strengths. INVX1 is the unit inverter; INVX2 has twice the drive. INVXL has the same area as the unit inverter but uses smaller transistors to reduce power consumption on noncritical paths. The X12-X20 inverters are built from three stages of smaller inverters to give high drive strength and low input capacitance at the expense of greater parasitic delay.

From the datasheet, we see the unit inverter has an input capacitance of 3.6 fF. The rising and falling delays are specified separately. We will develop a notation for different

delays in Section 6.2.1.5, but will use the average delay for now. The average *intrinsic* or parasitic delay is (25.3 + 14.6)/2 = 20.0 ps. The slope of the delay vs. load capacitance curve is the average of the rising and falling K_{load} values. An inverter driving a fanout of h will thus have a delay of

$$t_{pd} = 20.0\,\text{ps} + \left(3.6\frac{\text{fF}}{\text{gate}}\right)(h\ \text{gates})\left(\frac{4.53+2.37}{2}\frac{\text{ns}}{\text{pF}}\right) = (20.0+12.4h)\,\text{ps} \qquad (4.20)$$

The slope of the delay vs. fanout curve indicates τ = 12.4 ps and the y-intercept indicates p_{inv} = 20.0 ps, or 1.61 in normalized terms.

By a similar calculation, we find the X1 2-input NAND gate has an average delay from the inner (A) input of

$$t_{pd} = \left(\frac{31.3+19.5}{2}\right)\text{ps} + \left(4.2\frac{\text{fF}}{\text{gate}}\right)(h\ \text{gates})\left(\frac{4.53+2.84}{2}\frac{\text{ns}}{\text{pF}}\right) = (25.4+15.5h)\,\text{ps} \qquad (4.21)$$

Thus, the parasitic delay is 2.05 and the logical effort is 1.25. The parasitic delay from the outer (B) input is slightly higher, as expected. The parasitic delay and logical effort of the X2 and X4 gates are similar, confirming our model that logical effort should be independent of gate size for gates of reasonable sizes.

4.4 Power Dissipation

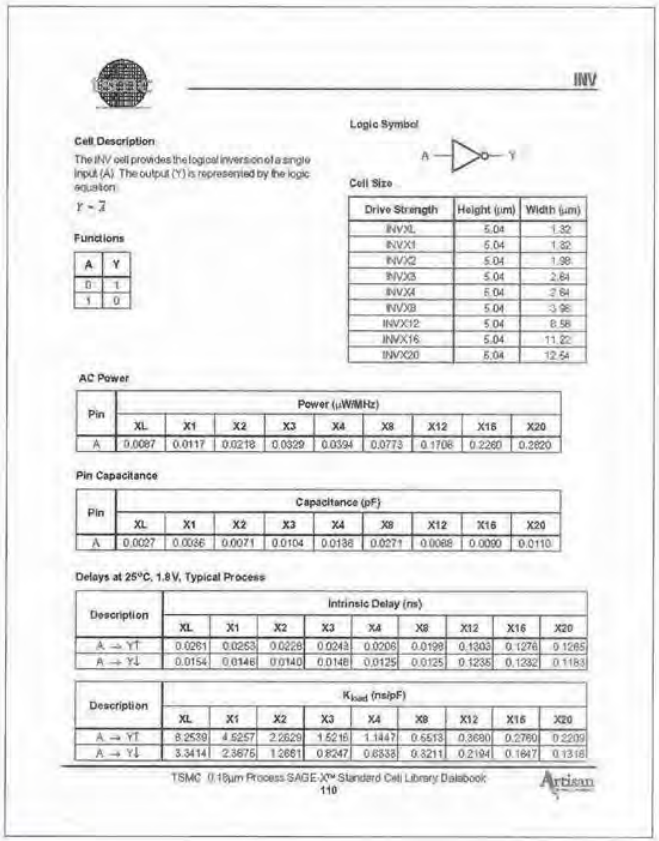
Static CMOS gates are very power-efficient because they dissipate nearly zero power while idle. For much of the history of CMOS design, power was a secondary consideration behind speed and area for many chips. As transistor counts and clock frequencies have increased, power consumption has skyrocketed and now is a primary design constraint.

We begin by reviewing some definitions. The *instantaneous power* $P(t)$ drawn from the power supply is proportional to the supply current $i_{DD}(t)$ and the supply voltage V_{DD}

$$P(t) = i_{DD}(t)V_{DD} \qquad (4.22)$$

The *energy* consumed over some time interval T is the integral of the instantaneous power

$$E = \int_0^T i_{DD}(t)V_{DD}\,dt \qquad (4.23)$$



INV

Cell Description

The INV cell provides the logical inversion of a single input (A). The output (Y) is represented by the logic equation:

$Y = \overline{A}$

Logic Symbol

Functions

A	Y
0	1
1	0

Cell Size

Drive Strength	Height (µm)	Width (µm)
INVXL	5.04	1.32
INVX1	5.04	1.32
INVX2	5.04	1.98
INVX3	5.04	2.64
INVX4	5.04	2.64
INVX8	5.04	3.96
INVX12	5.04	6.58
INVX16	5.04	11.22
INVX20	5.04	12.54

AC Power

Pin	Power (µW/MHz)								
	XL	X1	X2	X3	X4	X8	X12	X16	X20
A	0.0087	0.0117	0.0218	0.0329	0.0394	0.0773	0.1708	0.2260	0.2820

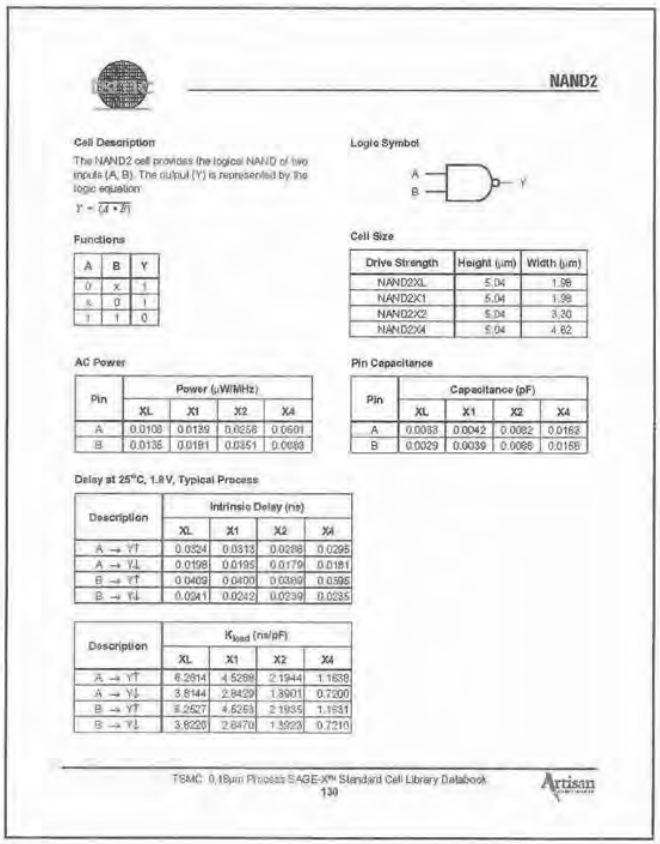
Pin Capacitance

Pin	Capacitance (pF)								
	XL	X1	X2	X3	X4	X8	X12	X16	X20
A	0.0027	0.0036	0.0071	0.0104	0.0138	0.0271	0.0668	0.0890	0.1110

Delays at 25°C, 1.8V, Typical Process

Description	Intrinsic Delay (ns)								
	XL	X1	X2	X3	X4	X8	X12	X16	X20
A → Y↑	0.0261	0.0253	0.0229	0.0243	0.0206	0.0198	0.1303	0.1276	0.1265
A → Y↓	0.0154	0.0146	0.0140	0.0148	0.0125	0.0125	0.1235	0.1232	0.1183

Description	K_{load} (ns/pF)								
	XL	X1	X2	X3	X4	X8	X12	X16	X20
A → Y↑	8.2539	4.5257	2.2629	1.5216	1.1447	0.5513	0.3690	0.2760	0.2209
A → Y↓	3.3414	2.3675	1.2661	0.8247	0.6338	0.3211	0.2194	0.1647	0.1318



NAND2

Cell Description

The NAND2 cell provides the logical NAND of two inputs (A, B). The output (Y) is represented by the logic equation:

$Y = \overline{(A \cdot B)}$

Logic Symbol

Functions

A	B	Y
0	X	1
X	0	1
1	1	0

Cell Size

Drive Strength	Height (µm)	Width (µm)
NAND2XL	5.04	1.98
NAND2X1	5.04	1.98
NAND2X2	5.04	3.30
NAND2X4	5.04	4.62

AC Power

Pin	Power (µW/MHz)			
	XL	X1	X2	X4
A	0.0108	0.0139	0.0256	0.0501
B	0.0136	0.0181	0.0351	0.0689

Pin Capacitance

Pin	Capacitance (pF)			
	XL	X1	X2	X4
A	0.0033	0.0042	0.0082	0.0163
B	0.0029	0.0039	0.0086	0.0169

Delay at 25°C, 1.8V, Typical Process

Description	Intrinsic Delay (ns)			
	XL	X1	X2	X4
A → Y↑	0.0324	0.0313	0.0286	0.0295
A → Y↓	0.0198	0.0195	0.0179	0.0181
B → Y↑	0.0409	0.0400	0.0369	0.0395
B → Y↓	0.0241	0.0242	0.0239	0.0235

Description	K_{load} (ns/pF)			
	XL	X1	X2	X4
A → Y↑	8.2614	4.5298	2.1944	1.1638
A → Y↓	3.8144	2.8429	1.3901	0.7200
B → Y↑	8.2527	4.5253	2.1935	1.1631
B → Y↓	3.8220	2.8470	1.3923	0.7210

FIG 4.25 Artisan Components cell library datasheets. Reprinted with permission.

The *average power* over this interval is

$$P_{avg} = \frac{E}{T} = \frac{1}{T}\int_0^T i_{DD}(t)V_{DD}dt \tag{4.24}$$

Power dissipation in CMOS circuits comes from two components:

- Static dissipation due to
  - subthreshold conduction through OFF transistors
  - tunneling current through gate oxide
  - leakage through reverse-biased diodes
  - contention current in ratioed circuits
- Dynamic dissipation due to
  - charging and discharging of load capacitances
  - "short-circuit" current while both pMOS and nMOS networks are partially ON

$$P_{total} = P_{static} + P_{dynamic} \tag{4.25}$$

This section quantifies each of these components of power and discusses techniques to minimize power consumption.

4.4.1 Static Dissipation

Considering the static CMOS inverter shown in Figure 4.26, if the input = '0,' the associated nMOS transistor is OFF and the pMOS transistor is ON. The output voltage is V_{DD} or logic '1.' When the input = '1,' the associated nMOS transistor is ON and the pMOS transistor is OFF. The output voltage is 0 volts (GND). Note that one of the transistors is always OFF when the gate is in either of these logic states. Ideally, no current flows through the OFF transistor so the power dissipation is zero when the circuit is quiescent, i.e., when no transistors are switching. Zero quiescent power dissipation is a principle advantage of CMOS over competing transistor technologies. However, secondary effects including subthreshold conduction, tunneling, and leakage lead to small amounts of static current flowing through the OFF transistor. Assuming the leakage current is constant so instantaneous and average power are the same, the static power dissipation is the product of total leakage current and the supply voltage.

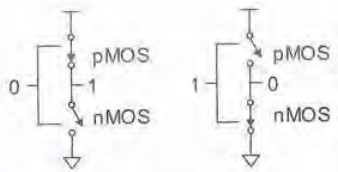


FIG 4.26 CMOS inverter model for static power dissipation evaluation

$$P_{static} = I_{static}V_{DD} \tag{4.26}$$

Example

A digital system in a 1.2 V 100 nm process [Parihar01] has 200 million transistors, of which 20 million are in logic gates and the remainder in memory arrays. The average logic transistor width is 12 λ and the average memory transistor width is 4 λ. The process has two threshold voltages and two oxide thicknesses. Subthreshold leakage for OFF devices is 20 nA/μm for low-threshold devices and 0.02 nA/μm for high-threshold devices. Gate leakage is 3 nA/μm for thin oxides and 0.002 nA/μm for thick oxides. Memories use low-leakage devices everywhere. Logic uses low-leakage devices in all but 20% of the paths that are most critical for performance. Diode leakage is negligible. Estimate the static power consumption. How would the power consumption change if the low-leakage devices were not available?

Solution: There are $(20 \cdot 10^6$ logic transistors$) \cdot (0.2) \cdot (12\,\lambda \cdot (0.05\,\mu m/\lambda) = 2.4 \cdot 10^6\,\mu m$ of high-leakage devices and $[(20 \cdot 10^6$ logic transistors$) \cdot (0.8) \cdot (12\,\lambda) + (180 \cdot 10^6$ memory transistors$) \cdot (4\,\lambda)] \cdot (0.05\,\mu m/\lambda) = 45.6 \cdot 10^6\,\mu m$ of low-leakage devices. All devices exhibit gate leakage. On average, half the transistors are OFF and contribute subthreshold leakage. Therefore, the total static current is $(2.4 \cdot 10^6\,\mu m) \cdot [(20\text{ nA}/\mu m)/2 + (3\text{ nA}/\mu m)] + (45.6 \cdot 10^6\,\mu m) \cdot [(0.02\text{ nA}/\mu m)/2 + (0.002\text{ nA}/\mu m)] = 32$ mA. Static power consumption is $(32\text{ mA}) \cdot (1.2\text{ V}) = 38$ mW. This is likely to be small compared to dynamic power consumption, yet large enough to limit the battery life of battery-powered systems on standby.

If low-leakage devices were not available, the total static current would be $(2.4 \cdot 10^6\,\mu m + 45.6 \cdot 10^6) \cdot [(20\text{ nA}/\mu m)/2 + (3\text{ nA}/\mu m)] = 624$ mA, for standby power of $(624\text{ mA}) \cdot (1.2\text{ V}) = 749$ mW.

According to EQ (2.34), OFF transistors still conduct a small amount of subthreshold current. As subthreshold current is exponentially dependent on threshold voltage, it is increasing dramatically as threshold voltages have scaled down.

SiO_2 is a very good insulator, so leakage current through the gate dielectric historically was very low. However, it is possible for electrons to tunnel across very thin insulators; the probability drops off exponentially with oxide thickness. Tunneling current becomes important for transistors around the 130 nm generation with gate oxides of 20Å or thinner.

There is also some small static dissipation due to reverse biased diode leakage between diffusion regions, wells, and the substrate, as shown for an inverter in Figure 2.19. Diode leakage is given by EQ (2.38). In modern processes, diode leakage is generally much smaller than the subthreshold or gate leakage and may be neglected.

In older processes, all three components of static power dissipation were small enough that CMOS was often said to consume "zero" DC power. Leakage power was of concern only to ultra-low-power systems. In 130 nm processes and beyond, the static power is rapidly becoming a primary design issue and vendors now provide leakage data, often in the form of nA/μm of gate length. Hand-held battery-powered devices typically require

Example

For a process with k'_p of 75 $\mu A/V^2$, V_{tp} = −0.4 V, and V_{DD} = 1.8 V, calculate the static power dissipation of a 32 word x 48-bit ROM that contains a 1:32 pseudo-nMOS row decoder and pMOS pull-ups on the 48-bit lines. The W/L ratio of the pMOS pull-ups is 1. Assume one of the word lines and 50% of the bitlines are high at any given time.

Solution: Each pMOS transistor dissipates power when the output is low.

$$I_{\text{pull-up}} = k'_p \frac{W}{L} \frac{\left(V_{DD} - |V_{tp}|\right)^2}{2} = 73\mu\text{A} \qquad (4.27)$$

$$P_{\text{pull-up}} = V_{DD} I_{\text{pull-up}} = 130\mu\text{W}$$

We expect to see 31 wordlines and 24 bitlines low, so the total static power is 130μW • (31 + 24) = 7.2 mW.

standby static currents in the 10's to 100's of μA. Eventually, static power dissipation may become comparable to dynamic power even for high-power systems.

Of course, static dissipation can occur in gates such as pseudo-nMOS gates where there is a direct path between power and ground. If such gates are used, this contention current must be factored into the total static power dissipation of the chip.

4.4.2 Dynamic Dissipation

The primary dynamic dissipation component is charging the load capacitance. Suppose a load C is switched between GND and V_{DD} at an average frequency of f_{sw}. Over any given interval of time T, the load will be charged and discharged Tf_{sw} times. Current flows from V_{DD} to the load to charge it. Current then flows from the load to GND during discharge. In one complete charge/discharge cycle, a total charge of $Q = CV_{DD}$ is thus transferred from V_{DD} to GND.

The average dynamic power dissipation is

$$P_{\text{dynamic}} = \frac{1}{T}\int_0^T i_{DD}(t) V_{DD}\, dt = \frac{V_{DD}}{T}\int_0^T i_{DD}(t)\, dt \qquad (4.28)$$

Taking the integral of the current over some interval T as the total charge delivered during that time, we simplify to

$$P_{\text{dynamic}} = \frac{V_{DD}}{T}\left[Tf_{sw} C V_{DD}\right] = C V_{DD}^2 f_{sw} \qquad (4.29)$$

Example

Our 200M transistor digital system from the example on page 99 uses static CMOS for the logic gates with an average activity factor of 0.1 The memory arrays are divided into banks and only the necessary bank is activated so the effective memory activity factor is 0.05. Assume transistors have a gate capacitance of about 2 fF/μm. Neglecting wire capacitance, estimate the dynamic power consumption per MHz of the system.

Solution: There are (20 • 10^6 logic transistors) • (12 λ) • (0.05 μm/λ) • (2 fF/μm) = 24 nF of logic transistors and (180 • 10^6 memory transistors) • (4 λ) • (0.05 μm/λ) • 2 fF/μm = 72 nF of memory transistors. The power consumption is [(0.1) • (24 • 10^{-9}) + (0.05) • (72 • 10^{-9})] • $(1.2)^2$ = 8.6 mW/MHz, or 8.6 W at 1 GHz.

Because most gates do not switch every clock cycle, it is often more convenient to express switching frequency f_{sw} as an activity factor α times the clock frequency f. Now the dynamic power dissipation may be rewritten as:

$$P_{\text{dynamic}} = \alpha C V_{DD}^2 f \tag{4.30}$$

A clock has an activity factor of α = 1 because it rises and falls every cycle. Most data has a maximum activity factor of 0.5 because it transitions only once each cycle. Static CMOS logic has been empirically determined to have activity factors closer to 0.1 because some gates maintain one output state more often than another and because real data inputs to some portions of a system often remain constant from one cycle to the next.

Because the input rise/fall time is greater than zero, both nMOS and pMOS transistors will be ON for a short period of time while the input is between V_{tn} and $V_{DD} - |V_{tp}|$. This results in an additional "short circuit" current pulse from V_{DD} to GND and typically increases power dissipation by about 10% [Veendrick84].

Short circuit power dissipation occurs as both pullup and pulldown networks are partially ON while the input switches [Veendrick84]. It increases as edge rates become slower because both networks are ON for more time. However, it decreases as load capacitance increases because with large loads the output only switches a small amount during the input transition, leading to a small V_{ds} across one of the transistors. Unless the input edge rate is much slower than the output edge rate, short circuit current is a small fraction (< 10%) of current to the load and can be ignored in hand calculations. It is good to use relatively crisp edge rates at the inputs to gates with wide transistors to minimize their short circuit current.

4.4.3 Low-power Design

Total power dissipation is the sum of the static and dynamic dissipation components. Dynamic dissipation has historically been far greater than static power when systems are active, and hence, static power is often ignored, although this will change as gate and sub-

threshold leakage increase. Many tools are available to assist with power estimation; these are discussed further in Sections 5.5.4 and 8.4.1.7.

Power dissipation has become extremely important to VLSI designers. For high-performance systems such as workstations and servers, dynamic power consumption per chip is often limited to about 150 W by the amount of heat that can be managed with air-cooled systems and cost-effective heatsinks. This number increases slowly with advances in heatsink technology and can be increased significantly with expensive liquid cooling, but has not kept pace with the growing power demands of systems. Therefore, performance may be limited by the inability to cool huge systems with power-hungry circuits operating at high speeds. For battery-based systems such as laptops, cell phones, and PDAs, power consumption sets the battery life of the product. In these systems, most or all of the switching activity may be stopped in an idle or "sleep" mode. Hence, in addition to dynamic power while active, static power consumption may limit the battery life while idle.

Many papers and books have been written on low-power design. Unfortunately, there are no "silver bullets"; low-power consumption is generally achieved by careful design. Power reduction techniques can be divided into those that reduce dynamic power and those that reduce static power.

4.4.3.1 Dynamic Power Reduction If a process is selected with sufficiently high threshold voltages and oxide thicknesses, static dissipation is small and dynamic dissipation usually dominates while the chip is active. EQ (4.30) shows that dynamic power is reduced by decreasing the activity factors, the switching capacitance, the power supply, or the operating frequency.

Activity factor reduction is very important. Static logic has an inherently low activity factor. Clocked nodes such as the clock network and the clock input to registers have an activity factor of 1 and are very power-hungry. Dynamic circuit families, described in Section 6.2.4, have clocked nodes and a high internal activity factor, so they are also costly in power. Clock gating can be used to stop portions of the chip that are idle; for example, a floating point unit can be turned off when executing integer code and a second level cache can be idled if the data is found in the primary cache. A large fraction of power is dissipated by the clock network itself, so entire portions of the clock network can be turned off where possible. The chip can also sense die temperature and cut back activity if the temperature becomes too high. A drawback of activity factor reduction is that if the system transitions rapidly from an idle mode with little switching to a fully active mode, a large di/dt spike will occur. This leads to inductive noise in the power supply network. Some systems throttle execution, limiting the number of functional units that go from idle to active in each cycle.

Device-switching capacitance is reduced by choosing small transistors. Minimum-sized gates can be used on non-critical paths. Although Logical Effort finds that the best stage effort is about 4, using a larger stage effort increases delay only slightly and greatly reduces transistor sizes. For example, buffers driving I/O pads or long wires may use a stage effort of 8–12 to reduce the buffer size. Interconnect switching capacitance is most

effectively reduced through careful floorplanning, placing communicating units near each other to reduce wire lengths.

Voltage has a quadratic effect on dynamic power. Therefore, choosing a lower power supply significantly reduces power consumption. As many transistors are operating in a velocity-saturated regime, the lower power supply may not reduce performance as much as first-order models predict. Voltage can be adjusted based on operating mode; for example, a laptop processor may operate at high voltage and high speed when plugged into an AC adapter, but at lower voltage and speed when on battery power. If the frequency and voltage scale down in proportion, a cubic reduction in power is achieved. For example, the laptop processor may scale back to 2/3 frequency and voltage to save 70% in power when unplugged.

Frequency can also be traded for power. For example, in a digital signal processing system primarily concerned with throughput, two multipliers running at half speed can replace a single multiplier at full speed. At first, this may not appear to be a good idea because it maintains constant power and performance while doubling area. However, if the power supply can also be reduced because the frequency requirement is lowered, overall power consumption goes down.

Commonly used metrics in low-power design are *power*, the *power-delay product*, and the *energy-delay product*. Power alone is a questionable metric because it can be reduced simply by computing more slowly. The power-delay (i.e., energy) product is also suspect because the energy can be reduced by computing more slowly at a lower supply voltage. The energy-delay product (i.e., power • delay^2) is less prone to such gaming.

Overall, the energy-delay product measured in Performance^2/Watt (where performance might be in units of SpecInt) normalized for process only varies by about a factor of two across a wide range of general-purpose microprocessor architectures [Gonzalez96]. This suggests that as long as wasteful practices are avoided, there is little you can do to general-purpose processors except trade the energy consumed by a computation against the delay of the computation. The big power gains are to be made not through tweaking of circuits but by reconsidering algorithms. For example, the Fast Fourier Transform requires far fewer arithmetic operations and hence less power than a Discrete Fourier Transform. Signal-processing systems using datapaths hardwired to a particular operation consume far less power than general-purpose processors delivering the same performance because the datapaths eliminate unnecessary control units.

4.4.3.2 Static Power Reduction Static power reduction involves minimizing I_{static}. Some circuit techniques such as analog current sources and pseudo-nMOS gates intentionally draw static power. They can be turned off when they are not needed.

Recall that the subthreshold leakage current for $V_{gs} < V_t$ is

$$I_{ds} = I_{ds0} e^{\frac{V_{gs} - V_t}{n v_T}} \left[1 - e^{\frac{-V_{ds}}{v_T}} \right] \tag{4.31}$$

$$V_t = V_{t0} - \eta V_{ds} + \gamma\left(\sqrt{\phi_s + V_{sb}} - \sqrt{\phi_s}\right) \tag{4.32}$$

where the η term describes drain-induced barrier lowering and the γ term describes the body effect. For any appreciable V_{ds}, the term in brackets approaches unity and can be discarded. The remaining term can be reduced by increasing the threshold voltage V_{t0}, reducing V_{gs}, reducing V_{ds}, increasing V_{sb}, or lowering the temperature.

Subthreshold leakage power is already a major problem for battery-powered designs in the 180 nm generation and will be growing exponentially as power supplies and threshold voltages are scaled down in future processes. Many low-power systems need high performance while active and low leakage while idle. The high-performance requirement entails relatively low thresholds, which contribute excessive leakage current in the idle mode. As mentioned in Section 4.4.1, selective application of multiple threshold voltages can maintain performance on critical paths with low-V_t transistors while reducing leakage on other paths with high-V_t transistors.

In low-power battery-operated devices, leakage specifications may be given at 40° C rather than 110° C because battery life is most important in the range of normal ambient temperatures.

Another way to control leakage is through the body voltage using the body effect. For example, low-V_t devices can be used and a *reverse body bias* (RBB) can be applied during idle mode to reduce leakage [Kuroda96]. Alternatively, higher-V_t devices can be used, and then a *forward body bias* (FBB) can be applied during active mode to increase performance [Narendra03]. As we will see in Section 4.7.3, threshold voltages vary from one die to another on account of manufacturing variations. An *adaptive body bias* (ABB) can compensate and achieve more uniform transistor performance despite the variations [Narendra99, Tschanz02]. In any case, the body bias should be kept to less than about 0.5 V. Too much reverse body bias leads to greater junction leakage through a mechanism called *band-to-band tunneling* [Keshavarzi01], while too much forward body bias leads to substantial current through the body to source diodes.

Applying a body bias requires additional power supply rails to distribute the substrate and well voltages. For example, a RBB scheme for a 1.8 V n-well process could bias the p-type substrate at $V_{BBn} = -0.4$ V and the n-well at $V_{BBp} = 2.2$ V. Figure 4.27 shows a schematic and cross-section of an inverter using body bias. In an n-well process, all nMOS transistors share the same p substrate and must use the same V_{BBn}. In a triple-well process, groups of transistors can use different p-wells isolated from the substrate and thus can use different body biases. The well and substrate carry little current, so the bias voltages are relatively easy to generate and distribute.

Alternatively, the source voltage can be raised in sleep mode. This has the double benefit of reducing V_{ds} (to increase V_t through reduced DIBL and to also reduce gate leakage) as well as increasing V_{sb} (to increase V_t through the body effect). However, the source does carry significant current, so generating a stable and adjustable source voltage rail is challenging.

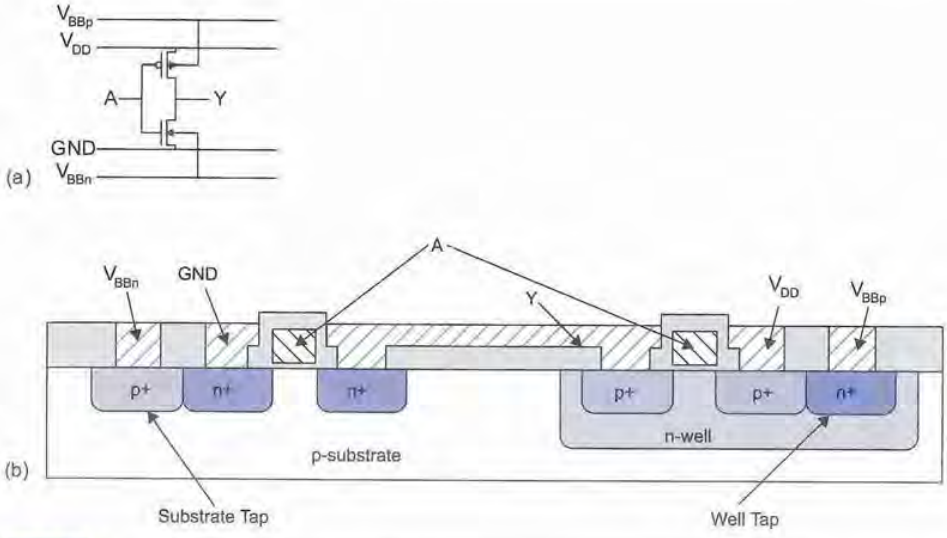


FIG 4.27 Body bias

Reducing V_{DD} in standby mode reduces the drain-induced barrier lowering contribution to leakage. It also decreases gate leakage in processes where that component is important. The supply should be maintained at a high enough level to preserve the state of the system [Clark02].

Yet another method of reducing idle leakage current in low-power systems is to turn off the power supply entirely. This could be done externally with the voltage regulator or internally with a series transistor. *Multiple Threshold CMOS* circuits (MTCMOS) use low-V_t transistors for computation and a high-V_t transistor as a switch to disconnect the power supply during idle mode, as shown in Figure 4.28 [Mutoh95][4]. The high-V_t device is connected between the true V_{DD} and the *virtual* V_{DDV} rails connected to the logic gates. The extra transistor increases the impedance between the true and virtual power supply, causing greater power supply noise and gate delay. Bypass capacitance between V_{DDV} and GND stabilizes the supply somewhat, but the capacitance is discharged each time V_{DDV} is disconnected, contributing to the power consumption. Even using a very wide high-V_t transistor, MTCMOS is only suited to systems with small power demands. The pMOS body should be tied to V_{DD} so both V_{DD} and V_{DDV} lines must be routed to all cells. MTCMOS uses carefully designed registers connected to the true supply rails to retain state during idle mode [Kao01].

[4]Do not confuse MTCMOS with the use of high-V_t transistors in noncritical gates and low-V_t transistors used in critical gates described earlier.

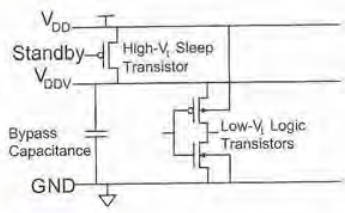


FIG 4.28 MTCMOS

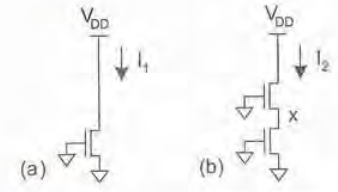


FIG 4.29 Leakage stack effect

The leakage through two series OFF transistors is much lower than that of a single transistor because of the *stack effect* [Ye98, Narendra01]. In Figure 4.29(a), the single transistor has a relatively low threshold because of drain-induced barrier lowering from the high drain voltage. In Figure 4.29(b), node x rises to about 100 mV. The threshold on the bottom transistor is higher because of the small drain voltage. The top transistor also turns off harder because of the negative V_{gs} and the body effect. The net result is that I_2 may be 10–20 times smaller than I_1. Low-power systems can take advantage of this stack effect to put gates with series transistors into a low-leakage sleep mode by applying an input pattern to turn off both transistors.

Silicon on Insulator (SOI) circuits are attractive for low-leakage designs because they have a sharper subthreshold current rolloff (smaller n in EQ (4.31)). SOI will be discussed further in Section 6.7.

4.5 Interconnect

The wires linking transistors together are called *interconnect* and play a major role in the performance of modern systems. Figure 4.30 shows a pair of adjacent wires. The wires have width w, length l, thickness t, and spacing of s from their neighbors and have a dielectric of height h between them and the conducting layer below. The sum of width and spacing is called the wire *pitch*. The thickness to width ratio t/w is called the *aspect ratio*. The dielectric is made of SiO_2 or a low-k material. In the early days of VLSI, transistors were relatively slow. Wires were wide and thick and thus had low resistance. Under those circumstances, wires could be treated as ideal equipotential nodes with lumped capacitance. In modern VLSI processes, transistors switch much faster. Meanwhile, wires have become narrower, driving up their resistance to the point that in many paths the wire RC delay exceeds gate delay. Moreover, the wires are packed very closely together and thus a large fraction of their capacitance is to their neighbors. When one wire switches, it tends to affect its neighbor through capacitive coupling; this effect is called *crosstalk*. On-chip interconnect inductance had been negligible but is now becoming a factor for systems with fast edge rates and closely packed busses. Considering all of these factors, circuit design is now as much about engineering the wires as the transistors that sit underneath.

Early CMOS processes had a single metal layer and for many years only two or three layers were available, but with advances in chemical-mechanical polishing it became far more practical to manufacture many metal layers. A 180 nm process typically has about six to eight metal layers and the layer count has been increasing at a rate of about one per generation. Figure 4.31 shows a cross-section of the Intel 180 nm process metal stack [Yang98]. Metal1 wires are thin and built on a tight pitch to provide dense routing within a cell; observe that the pitch is less than 6 λ, while conservative scalable CMOS rules would dictate an 8 λ pitch. Their resistance is high, but that is acceptable because the wires tend to be short. Top-level metal wires are thicker and built on a wide pitch.

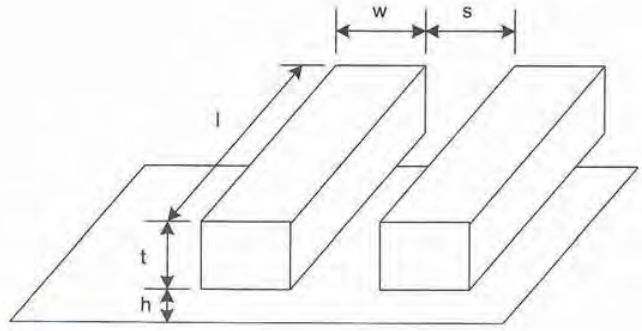


FIG 4.30 Interconnect geometry

Layer	t (nm)	w (nm)	s (nm)	AR
6	1720	860	860	2.0
	1000			
5	1600	800	800	2.0
	1000			
4	1080	540	540	2.0
	700			
3	700	320	320	2.2
	700			
2	700	320	320	2.2
	700			
1	480	250	250	1.9
	800			

Substrate

FIG 4.31 Layer stack for 6-metal Intel 180 nm process

This low-resistance layer is useful for power, ground, clock, and critical signal routing. In the Intel process, the intermediate layers gradually increase in width and pitch, although in many processes the intermediate layers are uniform. The width and spacing are typically comparable.

4.5.1 Resistance

The resistance of a uniform slab of conducting material can be expressed as

$$R = \frac{\rho}{t}\frac{l}{w} \tag{4.33}$$

where ρ is the resistivity[5]. This expression can be rewritten as

$$R = R_{\square}\frac{l}{w} \tag{4.34}$$

where $R_{\square} = \rho/t$ is the *sheet resistance* and has units of Ω/square. Note that a square is a dimensionless quantity corresponding to a slab of equal length and width. This is convenient because resistivity and thickness are characteristics of the process outside the control of the circuit designer and can be abstracted away into the single sheet resistance parameter.

To obtain the resistance of a conductor on a layer, multiply the sheet resistance by the ratio of length to width of the conductor. For example, the resistance of the two shapes in Figure 4.32 are equal because the length-to-width ratio is the same even though the sizes are different. Nonrectangular shapes can be decomposed into simpler regions for which the resistance is calculated [Horowitz83].

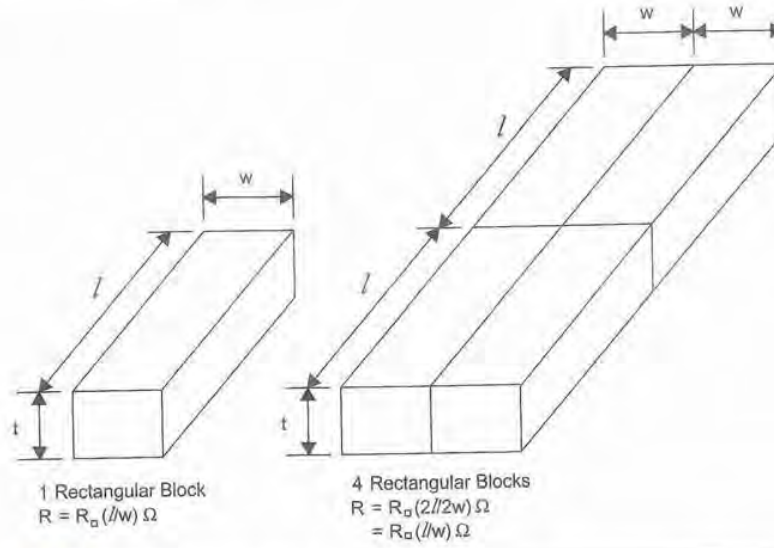


FIG 4.32 Two conductors with equal resistance

[5] ρ is used to indicate both resistivity and best stage effort. The meaning should be clear from context.

Table 4.6 shows bulk electrical resistivities of pure metals [Bakoglu90]. The sheet resistance of thin metal films used in wires tends to be slightly higher, e.g., 2.6 μΩ • cm for Cu and 3.5–4.0 μΩ • cm for Al. Most processes prior to the 180 nm generation use aluminum wires. Modern processes often use copper to reduce the resistivity and also to obtain better electromigration characteristics (see Section 4.8.2). Unfortunately, copper must be surrounded by a lower-conductivity diffusion barrier that effectively reduces the wire cross-sectional area and hence raises the resistance. Aluminum does not require such a barrier and thus may actually offer lower resistance for very narrow wires in the future. Electron surface scattering effects in thin conductors also result in somewhat higher resistance for on-chip interconnect than simple bulk resistivity would predict.

Table 4.6 Bulk resistivity of pure metals at 22° C

Metal	Resistivity (μΩ • cm)
Silver (Ag)	1.6
Copper (Cu)	1.7
Gold (Au)	2.2
Aluminum (Al)	2.8
Tungsten (W)	5.3
Molybdenum (Mo)	5.3
Titanium (Ti)	43.0

Table 4.7 shows typical sheet resistances for the 180 nm process with aluminum interconnect. The upper layers of metal have lower resistivity because they are thicker. Metal resistance is determined by the material (usually Al or Cu). The resistivity of polysilicon, diffusion, and wells is significantly influenced by the doping levels. Polysilicon and diffusion are often silicided with $TiSi_2$ (see Section 3.2.8) to reduce the resistance. Interconnect resistance increases with temperature; this effect is particularly pronounced for wells and diffusion.

Contacts and vias also have a resistance associated with them that is dependent on the contacted materials and size of the contact. Typical values are 2–20 Ω. Multiple contacts should be used to form low-resistance connections. Because current crowding tends to occur at the periphery of contacts, design rules dictate multiple small contacts rather than a single large contact, as shown in Figure 4.33. When current turns at a right angle or reverses, a square array of contacts is generally required, while when the flow is in the same direction, fewer contacts can be used.

Table 4.7 Sheet resistances

Layer	Sheet Resistance (Ω /□)
Diffusion (silicided)	3-10
Diffusion (unsilicided)	50-200
Polysilicon (silicided)	3-10
Polysilicon (unsilicided)	50-400
Metal1	0.08
Metal2	0.05
Metal3	0.05
Metal4	0.03
Metal5	0.02
Metal6	0.02

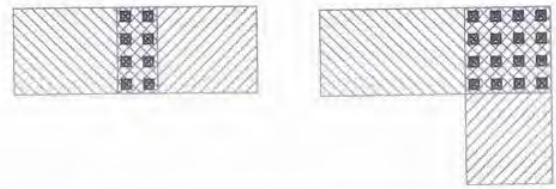


FIG 4.33 Multiple vias for low-resistance connections

4.5.2 Capacitance

An isolated wire over the substrate can be modeled as a conductor over a ground plane. The wire capacitance has two major components: the parallel plate capacitance of the bottom of the wire to ground and the fringing capacitance arising from fringing fields along the edge of a conductor with finite thickness. In addition, a wire adjacent to a second wire on the same layer can exhibit capacitance to that neighbor. These effects are illustrated in Figure 4.34. The classic parallel plate capacitance formula is

$$C = \frac{\varepsilon_{ox}}{h} wl \tag{4.35}$$

Note that oxides are often doped with phosphorous to trap ions before they damage transistors; this oxide has $\varepsilon_{ox} \approx 4.1\varepsilon_0$, as compared to $3.9\varepsilon_0$ for an ideal oxide or lower for low-k dielectrics.

The fringing capacitance is more complicated to compute and requires a numerical field solver for exact results. A number of authors have proposed approximations to this

calculation [Barke88, Ruehli73, Yuan82]. One intuitively appealing approximation treats a lone conductor above a ground plane as a rectangular middle section with two hemispherical end caps, as shown in Figure 4.35 [Yuan82]. The total capacitance is assumed to be the sum of a parallel plate capacitor of width $w - t/2$ and a cylindrical capacitor of radius $t/2$. This results in an expression for the capacitance that is accurate within 10% for aspect ratios less than 2 and $t \approx h$.

$$C = \varepsilon_{ox} l \left[\frac{w - \frac{t}{2}}{h} + \frac{2\pi}{\ln\left(1 + \frac{2h}{t} + \sqrt{\frac{2h}{t}\left(\frac{2h}{t} + 2\right)}\right)} \right] \tag{4.36}$$

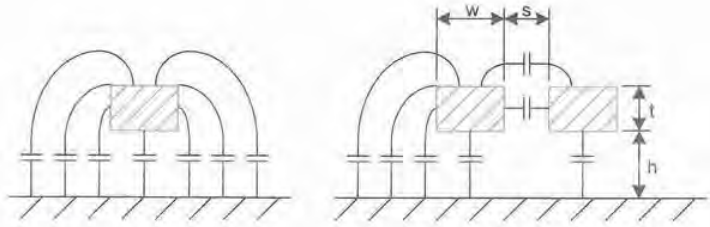


FIG 4.34 Effect of fringing fields on capacitance

An empirical formula that is computationally efficient and relatively accurate is [Meijs84, Barke88]

$$C = \varepsilon_{ox} l \left[\frac{w}{h} + 0.77 + 1.06\left(\frac{w}{h}\right)^{0.25} + 1.06\left(\frac{t}{h}\right)^{0.5} \right] \tag{4.37}$$

which is good to 6% for aspect ratios less than 3.3.

These formulae do not account for neighbors on the same layer or higher layers. Capacitance interactions between layers can become quite complex in modern multilayer CMOS processes. A conservative upper bound on capacitance can be obtained assuming that the layers above and below the conductor of interest are solid ground planes. Similarly, a lower bound can be obtained assuming there are no other conductors in the system except the substrate. The upper bound can be used for propagation delay and power estimation while the lower bound can be used for contamination delay calculations before layout information is available. A

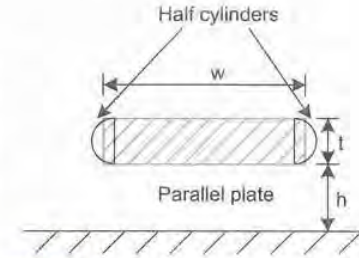


FIG 4.35 Yuan & Trick capacitance model including fringing fields

cross-section of the model used for capacitance upper bound calculations is shown in Figure 4.36. The total capacitance of the conductor of interest is the sum of its capacitance to the layer above, the layer below, and the two adjacent conductors. If the layers above and below are not switching[6], they can be modeled as ground planes and this component of capacitance is called C_{gnd}. Theoretically, wires will have some capacitance to further neighbors, but in practice this capacitance is normally small enough to ignore because most electric fields terminate on the nearest conductors.

$$C_{gnd} = C_{bot} + C_{top}$$
$$C_{total} = C_{gnd} + 2C_{adj} \tag{4.38}$$

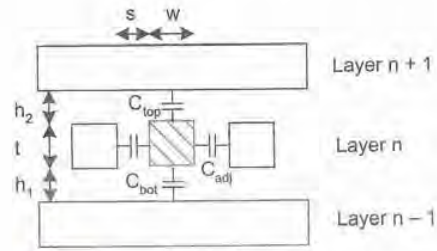


FIG 4.36 Multilayer capacitance model

The capacitances can be computed by generating a table of data with a field solver such as FastCap [Nabors92] or HSPICE. Tables can be generated for different widths and spacings on each layer and you can interpolate between entries if necessary. For example, Table 4.8 tabulates capacitance for the layer stack from Figure 4.31 using a SiOF low-k dielectric with $\varepsilon_{ox} = 3.55\varepsilon_0$ and assuming solid planes of metal above and below. The width w and spacing s indicate multiples of the minimum allowable for the particular metal layer. Three columns for each metal layer indicate the capacitance to each adjacent neighbor, to ground (i.e., the planes above and below, which are assumed on average not to be moving), and total (EQ(4.38)). The capacitance to neighbors accounts for more than 50% of the capacitance for narrow wires. We will see in Section 4.5.4 that coupling to these neighbors has a significant effect on delay and signal integrity and limits the acceptable aspect ratios. The capacitance for metal2 and metal3 lines should be equal because they have the same geometry; the minor variations reflect finite numerical precision in the HSPICE field solver. Table 4.9 shows the same information assuming no metal on other planes. The overall capacitance is slightly smaller because there are no nearby planes for fields to terminate on, but the capacitance to the adjacent neighbors is higher because many of the fringing fields now terminate on the neighbors instead.

[6]Or at least consist of a large number of orthogonal conductors that on average cancel each other's switching activities.

Table 4.8 Capacitance table for 180 nm process (aF/μm) with metal planes above and below

w	s	Metal1			Metal2			Metal3			Metal4			Metal5			Metal6		
		C_{adj}	C_{gnd}	C_{tot}	C_{adj}	C_{gnd}	C_{tot}	C_{adj}	C_{gnd}	C_{tot}	C_{adj}	C_{gnd}	C_{tot}	C_{adj}	C_{gnd}	C_{tot}	C_{adj}	C_{gnd}	C_{tot}
1	1	84	43	210	88	57	232	88	57	232	77	78	232	73	94	240	82	64	227
1	1.5	57	52	166	58	68	184	58	68	184	49	93	191	45	112	202	54	76	184
1	2	42	60	144	42	78	162	42	78	162	33	106	173	30	126	186	39	85	164
1	∞	0	112	112	0	132	132	0	132	132	0	154	154	0	167	167	0	137	137
1.5	1	85	53	224	89	70	248	89	70	248	77	99	253	73	119	266	83	80	246
1.5	1.5	58	62	178	59	82	199	59	82	199	49	114	212	45	137	227	56	91	202
1.5	2	43	70	156	42	92	177	42	92	177	34	127	194	30	151	211	41	101	182
1.5	∞	0	123	123	0	147	147	0	147	147	0	174	174	0	192	192	0	154	154
2	1	86	63	236	89	84	263	89	84	263	77	119	274	73	144	291	84	95	264
2	1.5	59	72	190	59	96	214	59	96	214	49	134	232	45	162	252	57	107	220
2	2	44	80	167	43	106	191	43	106	191	34	147	215	30	176	236	41	117	199
2	∞	0	134	134	0	162	162	0	162	162	0	195	195	0	217	217	0	171	171
3	1	87	83	258	90	113	292	90	113	292	77	160	315	73	194	341	86	127	298
3	1.5	60	92	211	59	124	243	59	124	243	49	175	273	45	212	302	58	138	254
3	2	44	101	189	43	135	220	43	135	220	34	188	256	30	226	286	42	148	233
3	∞	0	155	155	0	191	191	0	191	191	0	236	236	0	269	269	0	204	204
4	1	87	104	279	90	141	320	90	141	320	77	201	355	73	245	390	86	159	331
4	1.5	60	113	232	59	153	272	59	153	272	49	216	314	45	262	352	58	170	287
4	2	44	122	210	43	164	249	43	164	249	34	229	296	30	276	336	43	180	266
4	∞	0	176	176	0	219	219	0	219	219	0	278	278	0	321	321	0	237	237
6	1	87	146	320	89	199	377	89	199	377	77	284	437	72	345	490	86	223	396
6	1.5	59	155	274	59	210	328	59	210	328	49	299	396	45	362	451	58	235	351
6	2	44	163	252	42	221	306	42	221	306	33	312	378	30	376	436	43	245	331
6	∞	0	218	218	0	276	276	0	276	276	0	360	360	0	422	422	0	301	301

Figure 4.37 plots total capacitance of a metal2 line from the same process as a function of width for various spacings. For an isolated wire above the substrate, the capacitance is strongly influenced by spacing between conductors. For a wire sandwiched between metal1 and metal3 planes, the capacitance is higher and is more sensitive to the width (determining parallel plate capacitance) but less sensitive to spacing once the spacing is significantly greater than the wire thickness. In either case, the y-intercept is greater than zero so doubling the width of a wire results in less than double the total capacitance. Tight-pitch metal lines have a capacitance of roughly 0.2 fF/μm.

Table 4.9 Capacitance table for 180 nm process (aF/μm) with substrate below and nothing above

w	s	Metal1			Metal2			Metal3			Metal4			Metal5			Metal6		
		C_{adj}	C_{gnd}	C_{tot}	C_{adj}	C_{gnd}	C_{tot}	C_{adj}	C_{gnd}	C_{tot}	C_{adj}	C_{gnd}	C_{tot}	C_{adj}	C_{gnd}	C_{tot}	C_{adj}	C_{gnd}	C_{tot}
1	1	89	28	206	101	19	220	96	16	208	102	15	219	96	15	208	97	14	207
1	1.5	63	33	159	73	22	167	70	18	158	75	17	166	70	17	158	71	15	158
1	2	49	37	136	58	24	140	57	20	133	60	18	138	57	19	132	58	17	132
1	∞	0	87	87	0	70	71	0	60	61	0	59	60	0	60	60	0	55	55
1.5	1	92	33	217	104	21	230	100	17	218	106	16	229	100	17	217	101	15	217
1.5	1.5	66	38	170	76	24	176	74	19	167	78	18	174	74	19	167	75	16	166
1.5	2	52	42	145	61	27	148	60	21	140	63	20	146	60	21	140	61	18	139
1.5	∞	0	94	94	0	75	75	0	64	64	0	62	62	0	63	63	0	58	58
2	1	94	38	227	107	24	238	103	19	225	110	18	237	103	18	225	104	16	224
2	1.5	68	43	178	78	26	183	76	21	174	81	19	181	77	20	173	78	17	173
2	2	53	47	154	63	29	154	62	22	147	66	21	152	62	22	146	63	19	145
2	∞	0	100	100	0	78	79	0	67	67	0	65	65	0	66	66	0	60	60
3	1	98	48	243	111	29	251	108	22	237	114	20	249	108	21	237	109	18	236
3	1.5	71	52	194	82	31	195	81	24	185	85	22	192	81	23	184	82	20	183
3	2	56	57	169	66	34	166	66	25	157	69	24	162	66	25	156	67	21	155
3	∞	0	112	112	0	86	86	0	72	72	0	69	70	0	71	72	0	64	65
4	1	100	58	257	114	33	262	111	25	247	118	23	258	111	24	246	112	20	245
4	1.5	73	62	208	85	36	205	83	27	194	88	25	201	83	26	193	85	22	192
4	2	58	67	182	68	38	175	68	28	165	72	26	171	68	28	165	70	23	163
4	∞	0	124	124	0	92	93	0	77	77	0	74	75	0	76	76	0	68	69
6	1	102	77	282	118	43	279	115	31	262	122	28	272	115	31	260	116	25	258
6	1.5	76	82	233	88	46	222	87	33	208	92	30	215	87	33	207	89	26	204
6	2	61	86	207	72	48	191	72	35	179	76	31	184	72	34	178	73	28	175
6	∞	0	147	147	0	105	105	0	86	86	0	82	83	0	85	85	0	75	76

In practice, the layers above and below the conductor of interest are neither solid planes nor totally empty. One can extract capacitance more accurately by interpolating between these two extremes based on the density of metal on each level. [Chern92] gives formulae for this interpolation accurate to within 10%. However, if the wiring above and below is fairly dense (e.g., a bus on minimum pitch), it is well-approximated as a plane. Dense wire fill is added to many chips for mechanical stability and etch uniformity, making this approximation even more appropriate.

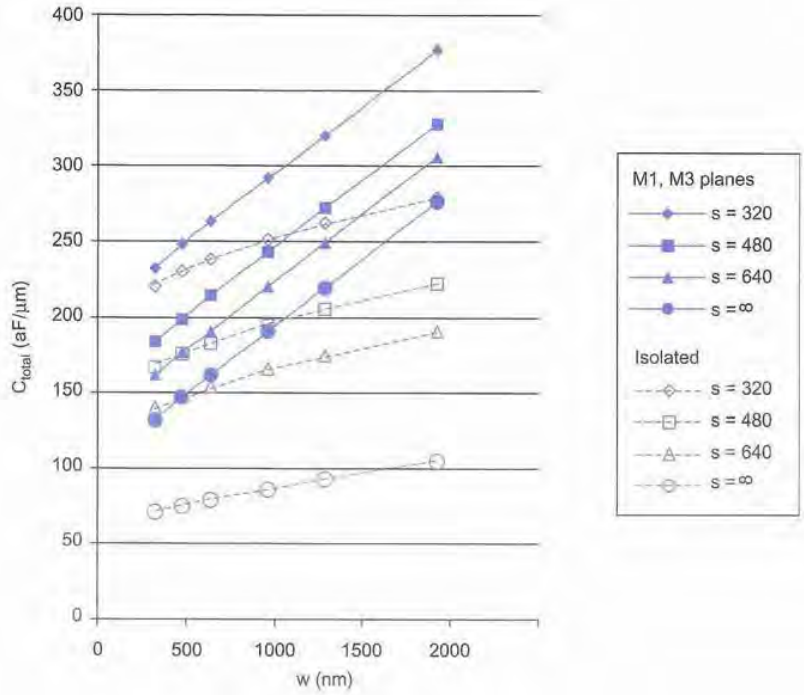


FIG 4.37 Capacitance of metal2 line as a function of width and spacing

4.5.3 Delay

Interconnect increases circuit delay for two reasons. First, the wire capacitance adds loading to each gate. Second, long wires have significant resistance that contributes distributed RC delay or *flight time*. It is straightforward to add wire capacitance to the Elmore delay calculations of Section 4.2.1, so in this section we focus on the RC delay.

The distributed resistance and capacitance of a wire can be approximated with a number of lumped elements. Three standard approximations are the L-model, π-model, and T-model, so-named because of their shape. Figure 4.38 shows how a distributed RC circuit is equivalent to N distributed RC segments of proportionally less resistance and capacitance, and how these segments

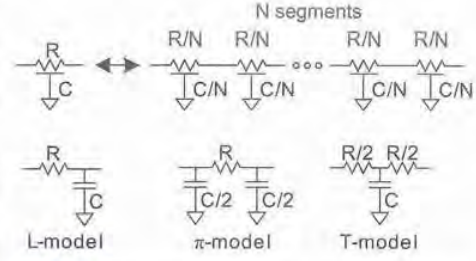


FIG 4.38 Lumped approximation to distributed RC circuit

Example

Consider a 5 mm long, 0.32 μm wide metal2 wire in a 180 nm process. The sheet resistance is 0.05 Ω/□ and the capacitance is 0.2 fF/μm. Construct a 3-segment π-model for the wire.

Solution: The wire is 5000 μm/0.32 μm = 15625 squares in length. The total resistance is (0.05 Ω/□) • (15625□) = 781 Ω. The total capacitance is (0.2 fF/μm) • (5000 μm) = 1 pF. Each π-segment has one-third of this resistance and capacitance. The π-model is shown in Figure 4.39(a); adjacent capacitors can be merged as shown in Figure 4.39(b).

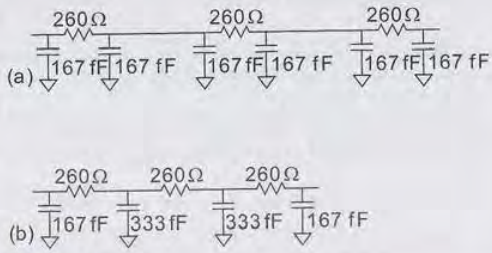


FIG 4.39 3-segment π-model for wire

Example

A 10x unit-sized inverter drives a 2x inverter at the end of the 5 mm wire from the previous example. The gate capacitance is C = 2 fF/μm and the effective resistance is R = 2.5 kΩ • μm for nMOS transistors. Estimate the propagation delay using the Elmore delay model; neglect diffusion capacitance.

Solution: A unit inverter has a 4 λ = 0.36 μm wide nMOS transistor and an 8 λ = 0.72 μm wide pMOS transistor. Hence, the unit inverter has an effective resistance of (2.5 kΩ • μm)/(0.36 μm) = 6.9 kΩ and a gate capacitance of (0.36 μm + 0.72 μm) • (2 fF/μm) = 2 fF. Larger inverters have proportionally more capacitance and less resistance. Figure 4.40 shows an equivalent circuit for the system using a single-segment π-model. The Elmore delay is t_{pd} = (690 Ω) • (500 fF) + (690 Ω + 781 Ω) • (500 fF + 4 fF) = 1.1 ns. The capacitance of the long wire dominates the delay; the capacitance of the 2x inverter is negligible in comparison.

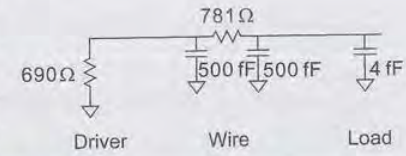


FIG 4.40 Equivalent circuit for example

can be modeled with lumped elements. As the number of segments approaches infinity, the lumped approximation will converge with the true distributed circuit. The L-model is a very poor choice because a large number of segments are required for accurate results. The π-model is much better; three segments are sufficient to give results accurate to 3% [Sakurai83]. The T-model is comparable to the π-model, but produces circuits with more nodes that are slower to solve by hand or with a circuit simulator. Therefore, it is common practice to model long wires with a 3- or 4-segment π-model for simulation.

These results make sense because the Elmore delay of a single-segment L-model is RC while the Elmore delay of a single-segment π- or T-model is $RC/2$. Single-segment π-models are a reasonable approximation for hand calculations.

Because both wire resistance and wire capacitance increase with length, wire delay grows quadratically with length. Using thicker and wider wires, lower-resistance metals such as copper, and lower-dielectric constant insulators helps, but long wires nevertheless often have unacceptable delay. Section 4.6.4 describes how repeaters can be used to break a long wire into multiple segments such that the overall delay becomes a linear function of length.

Polysilicon and diffusion wires (sometimes called *runners*) have high resistance, even if silicided. Diffusion also has very high capacitance. Do not use diffusion. Use polysilicon sparingly, usually in latches and flip-flops.

4.5.4 Crosstalk

As reviewed in Figure 4.41, wires have capacitance to their adjacent neighbors as well as to ground. When wire A switches, it tends to bring its neighbor B along with it on account of capacitive coupling, also called *crosstalk*. If B is supposed to switch simultaneously, this may increase or decrease the switching delay. If B is not supposed to switch, crosstalk causes noise on B. We will see that the impact of crosstalk depends on the ratio of C_{adj} to the total capacitance. Note that the load capacitance is included in the total, so for short wires and large loads, the load capacitance dominates and crosstalk is unimportant. Conversely, for long wires crosstalk is very important.

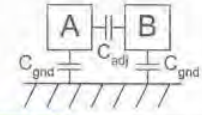


FIG 4.41 Capacitances to adjacent neighbor and to ground

4.5.4.1 Crosstalk Delay Effects If both a wire and its neighbor are switching, the direction of the switching affects the amount of charge that must be delivered and the delay of the switching. Table 4.10 summarizes this effect. The charge delivered to the coupling capacitor is $Q = C_{adj}\Delta V$, where ΔV is the change in voltage between A and B. If A switches but B does not, $\Delta V = V_{DD}$. The total capacitance effectively seen by A is just the capacitance to ground and to B. If both A and B switch in the same direction, $\Delta V = 0$. Hence, no charge is required and C_{adj} is effectively absent for delay purposes. If A and B switch in the opposite direction, $\Delta V = 2V_{DD}$. Twice as much charge is required. Equivalently, the capacitor can be treated as being effectively twice as large switching through V_{DD}. This is analogous to the Miller effect discussed in Section 4.2.5.4. The *Miller Coupling Factor* (MCF)

describes how the capacitance to adjacent wires is multiplied to find the effective capacitance. Some designers use MCF = 1.5 as a statistical compromise when estimating propagation delays before layout information is available.

Table 4.10 Dependence of effective capacitance on switching direction

B	ΔV	$C_{eff(A)}$	MCF
constant	V_{DD}	$C_{gnd} + C_{adj}$	1
switching same direction as A	0	C_{gnd}	0
switching opposite to A	$2V_{DD}$	$C_{gnd} + 2C_{adj}$	2

Example

Each wire in a pair of 1 mm lines has capacitance of 0.1 fF/μm to ground and 0.1 fF/μm to its neighbor. Each line is driven by an inverter with a 1 kΩ effective resistance. Estimate the contamination and propagation delays of the path. Neglect parasitic capacitance of the inverter and resistance of the wires.

Solution: We find $C_{gnd} = C_{adj} = (0.1\ fF/\mu m) \cdot (1000\ \mu m) = 0.1\ pF$. The delay is RC_{eff}. The contamination delay is the minimum possible delay, which occurs when both wires switch in the same direction. In that case, $C_{eff} = C_{gnd}$ and the delay is $t_{cd} = (1k\Omega) \cdot (0.1\ pF) = 100$ ps. The propagation delay is the maximum possible delay, which occurs when both wires switch in opposite directions. In this case, $C_{eff} = C_{gnd} + 2C_{adj}$ and the delay is $t_{pd} = (1k\Omega) \cdot (0.3\ pF) = 300$ ps.

A conservative design methodology assumes neighbors are switching when computing propagation and contamination delays (MCF = 2 and 0, respectively). This leads to a wide variation in the delay of wires. A more aggressive methodology tracks the time window during which each signal can switch. Thus, switching neighbors must be accounted for only if the potential switching windows overlap. Similarly, the direction of switching can be considered. For example, dynamic gates described in Section 6.2.4 precharge high and then fall low during evaluation. Thus, a dynamic bus will never see opposite switching during evaluation.

4.5.4.2 Crosstalk Noise Effects Suppose wire A switches while B is supposed to remain constant. This introduces noise as B partially switches. We call A the *aggressor* or *perpetrator* and B the *victim*. If the victim is floating, we can model the circuit as a capacitive volt-

age divider to compute the victim noise, as shown in Figure 4.42. $\Delta V_{aggressor}$ is normally V_{DD}.

$$\Delta V_{victim} = \frac{C_{adj}}{C_{gnd-v} + C_{adj}} \Delta V_{aggressor} \tag{4.39}$$

If the victim is actively driven, the driver will supply current to oppose and reduce the victim noise. We model the drivers as resistors, as shown in Figure 4.43. The peak noise becomes dependent on the time constant ratio k of the aggressor to the victim [Ho01]:

$$\Delta V_{victim} = \frac{C_{adj}}{C_{gnd-v} + C_{adj}} \frac{1}{1+k} \Delta V_{aggressor} \tag{4.40}$$

where

$$k = \frac{\tau_{aggressor}}{\tau_{victim}} = \frac{R_{aggressor}\left(C_{gnd-a} + C_{adj}\right)}{R_{victim}\left(C_{gnd-v} + C_{adj}\right)} \tag{4.41}$$

Figure 4.44 shows simulations of coupling when the aggressor is driven with a unit inverter; the victim is undriven or driven with an inverter of half, equal, or twice the size of the aggressor; and $C_{adj} = C_{gnd}$. Observe that when the victim is floating, the noise remains indefinitely. When the victim is driven, the driver restores the victim. Larger (faster) drivers oppose the coupling sooner and result in noise that is a smaller percentage of the supply voltage. Note that during the noise event the victim transistor is in its linear region while the aggressor is in saturation. For equal-sized drivers, this means $R_{aggressor}$ is two to four times R_{victim}, with greater ratios arising from more velocity saturation [Ho01]. In general, EQ (4.40) is conservative, especially when wire resistance is included [Vittal99]. It is often used to flag nets where coupling can be a problem; then simulations can be performed to calculate the exact coupling noise. Coupling noise is of greatest importance on weakly driven nodes where $k < 1$.

We have only considered the case of a single neighbor switching. When both neighbors switch, the noise will be twice as great. We have also modeled the layers above and below as AC ground planes, but wires on these layers are likely to be switching. For a long line, you can expect about as many

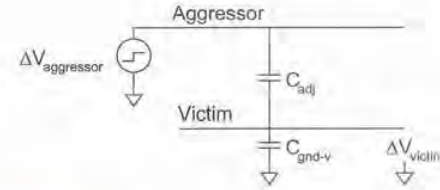


FIG 4.42 Coupling to floating victim

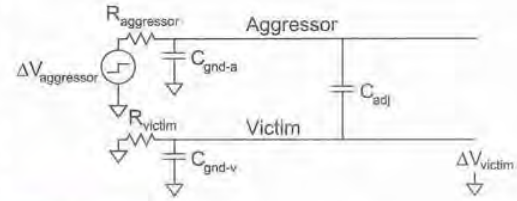


FIG 4.43 Coupling to driven victim

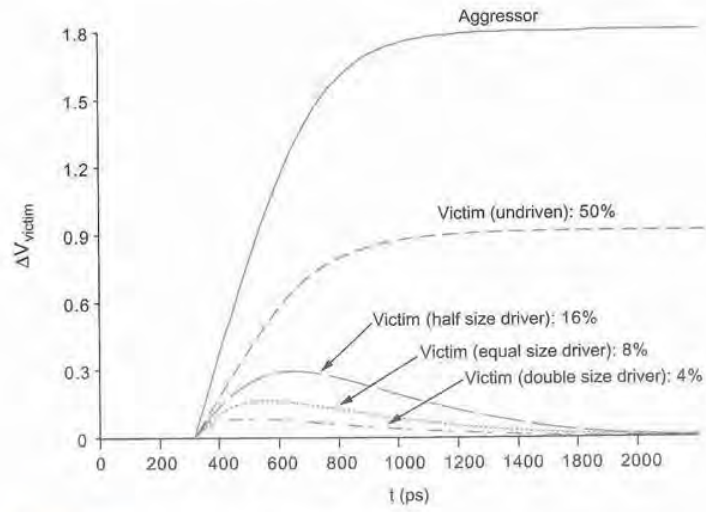


FIG 4.44 Waveforms of coupling noise

lines switching up and switching down, giving no net contribution to delay or noise. However, a short line running over a 64-bit bus in which all 64 bits are simultaneously switching from 0 to 1 will be strongly influenced by this switching.



4.5.5 Inductance

Most design tools consider only interconnect resistance and capacitance. Inductance is beginning to be important for accurately modeling on-chip power grids, clock networks, and wide busses. While the industry still has a very limited understanding of inductive effects, some of the key issues are introduced in this section.

Although we generally discuss current flowing from a gate output to charge or discharge a load capacitance, current really flows in loops. The return path for a current loop is usually the power or ground network; at the frequencies of interest, the power supply is an "AC ground" because the bypass capacitance forms a low-impedance path between V_{DD} and GND. Currents flowing around a loop generate a magnetic field proportional to the area of the loop and the amount of current. Changing the current requires supplying energy to change the magnetic field. This means that changing currents induce a voltage proportional to the rate of change. The constant of proportionality is called the inductance, L[7].

[7] L is used to indicate both inductance and transistor channel length. The meaning should be clear from context.

$$V = L\frac{dI}{dt} \tag{4.42}$$

Inductance and capacitance also set the speed of light in a medium. Even if the resistance of a wire is zero leading to zero RC delay, the speed of light flight-time along a wire of length with inductance and capacitance per unit length of L and C is

$$t_{pd} = l\sqrt{LC} \tag{4.43}$$

If the current return paths are the same as the conductors on which electric field lines terminate, the signal velocity v is

$$v = \frac{1}{\sqrt{LC}} = \frac{1}{\sqrt{\varepsilon_{ox}\mu_0}} = \frac{c}{\sqrt{3.9}} \tag{4.44}$$

where μ_0 is the magnetic permeability of free space ($4\pi \cdot 10^{-7}$ H/m) and c is the speed of light in free space ($3 \cdot 10^8$ m/s). In other words, signals travel about half the speed of light. Using low-k (< 3.9) dielectrics raises this velocity. However, many signals have electric fields terminating on nearby neighbors, but currents returning in more distant power supply lines. This raises the inductance and reduces the signal velocity.

Changing magnetic fields in turn produce currents in other loops. This means that signals on one wire can inductively couple onto another; this is called *inductive crosstalk*.

The inductance of a conductor of length l and width w located a height h above a ground plane is approximately

$$L = l\frac{\mu_0}{2\pi}\ln\left(\frac{8h}{w} + \frac{w}{4h}\right) \tag{4.45}$$

assuming $w < h$ and thickness is negligible. Typical on-chip inductance values are in the range of 0.15–1.5 pH/μm depending on the proximity of the power or ground lines. (Wires near their return path have smaller current loops and lower inductance.)

Current flows along the path of lowest impedance $Z = R + j\omega L$. At high-frequency ω, impedance becomes dominated by inductance. The inductance is minimized if the current flows only near the surface of the conductor closest to the return path. This skin effect can reduce the effective cross-sectional area of thick conductors and raise the effective resistance at high frequency. The skin depth for a conductor is

$$\delta = \sqrt{\frac{2\rho}{\omega\mu_0}} \tag{4.46}$$

Example

Find the skin depth for signals with 100 ps edge rates on copper interconnect ($\rho = 1.7 \cdot 10^{-8}\,\Omega \cdot m$).

Solution: 100 ps edge rates correspond to energy at 1.67 GHz.

$$\delta = \sqrt{\frac{2\left(1.7\times10^{-8}\,\Omega\cdot m\right)}{\left(2\pi\cdot1.67\times10^{9}\,rad/s\right)\left(4\pi\times10^{-7}\,H/m\right)}} = 1.6\ \mu m \qquad (4.47)$$

Skin depth is not a major consideration at the time of writing, but will limit the benefit of thick wires at higher frequencies in the future.

The frequency of importance is the highest frequency with significant power in the Fourier transform of the signal. This is not the chip-operating frequency, but rather is associated with the faster edges. It can be approximated as

$$\omega = \frac{2\pi}{6t_{rf}} \qquad (4.48)$$

Extracting inductance in general is a three-dimensional problem and is extremely time-consuming for complex geometries. Inductance depends on the entire loop and therefore cannot be simply decomposed into sections in the way capacitance is. It is therefore impractical to extract the inductance from a chip layout. Instead, usually inductance is extracted using tools such as FastHenry [Kamon94] for simple test structures intended to capture the worst cases on the chip. This is only possible when the power supply network is highly regular. Power planes are ideal but require a large amount of metal resources. Dense power grids are usually the preferred alternative. Gaps in the power grid force current to flow around the gap, increasing the loop area and greatly increasing inductance. Moreover, large loops couple magnetic fields through other loops formed by conductors at a distance. Therefore, mutual inductive coupling can occur over a long distance, especially when the return path is far from the conductor. Incorporating inductance into simulations is also difficult. Instead, designers usually generate design rules that allow inductance to generally be ignored as long as the rules are followed.

Inductance has always been important for integrated circuit packages where the physical dimensions are large, as will be discussed in Section 12.2.3. On-chip inductance is important for wires where the speed of light flight time is longer than either the rise times of the circuits or the RC delay of the wire. Because speed of light flight time increases linearly and RC delay increases quadratically with length, we can estimate the set of wire lengths for which inductance is relevant [Ismail99].

Example

Consider a metal2 signal line with a sheet resistance of 0.05 Ω/□ and a width of 0.5 μm. The capacitance is 0.2 fF/μm and inductance is 0.5 pH/μm. Compute the velocity of signals on the line and plot the range of lengths over which inductance matters as a function of the rise time.

Solution: The velocity is

$$v = \frac{1}{\sqrt{LC}} = \frac{1}{\sqrt{(0.5\ \text{pH}/\mu\text{m})(0.2\ \text{fF}/\mu\text{m})}} = 10^8\ \text{m/s} = \tfrac{1}{3}c \qquad (4.49)$$

Note that this is 100 mm/ns or 1 mm / 10 ps. The resistance is (0.1 Ω/□) • (1□/0.5 μm) = 0.2 Ω/μm. Figure 4.45 plots the length of wires for which inductance is relevant against rise times. Above the horizontal line, wires greater than 500 μm are limited by RC delay rather than LC delay. To the right of the diagonal line, rise times are greater than the LC delay. Only in the region between these lines is inductance relevant to delay calculations. This region has very fast edge rates, so inductance is not very important to the delay of highly resistive signal lines at the time of this writ-

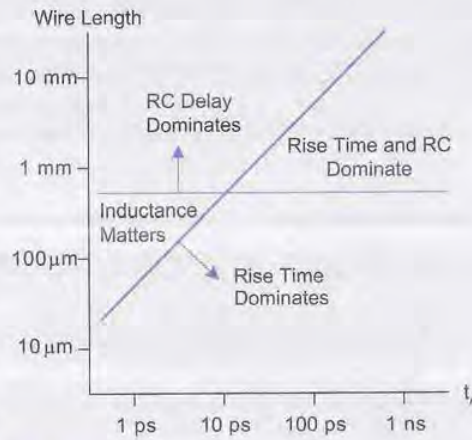


FIG 4.45 Wire lengths and edge rates for which inductance impacts delay

$$\frac{t_r}{2\sqrt{LC}} < l < \frac{2}{R}\sqrt{\frac{L}{C}} \qquad (4.50)$$

As the example illustrated, inductance will only be important to the delay of low-resistance signals such as wide clock lines or the power supply. As edge rates become faster, inductance will become relevant to a larger number of on-chip signals. Inductive crosstalk is also important for wide busses far away from their current return paths.

In power distribution networks, inductance means that if one portion of the chip requires a rapidly increasing amount of current, that charge must be delivered from nearby decoupling capacitors or supply pins; portions of the chip further away are unaware of the

changing current needs until a speed-of-light flight time has elapsed and hence will not supply current immediately. Adding inductance to the power grid simulation generally reveals greater supply noise than would otherwise be predicted. Power supplies will be discussed further in Section 12.3.

In wide, thick, upper-level metal lines, resistance and RC delay may be small. This pushes the horizontal line in Figure 4.45 upward, increasing the range of edge rates for which inductance matters. This is especially common for clock signals. Inductance tends to increase the propagation delay and sharpen the edge rate.

To see the effects of inductance, consider a 5 mm metal6 clock line above a metal5 ground plane driving a 2pF clock load. If its width is 4.8 μm (six times minimum), it has resistance of 4 Ω/mm, capacitance of 0.4 pF/mm, and inductance of 0.12 nH/mm. Figure 4.46 presents models of the clock line as a 5-stage π-model without (a) and with (b) inductance. Figure 4.46(c) shows the response of each model to an ideal voltage source with 80 ps rise time. The model including inductance shows a greater delay until the clock begins to rise because of the speed of light flight time. It also overshoots. However, the rising edge is sharper and the rise time is shorter. In some circumstances when the driver impedance is matched to the characteristic impedance of the wire, the sharper rising edge can actually result in a shorter propagation delay measured at the 50% point.

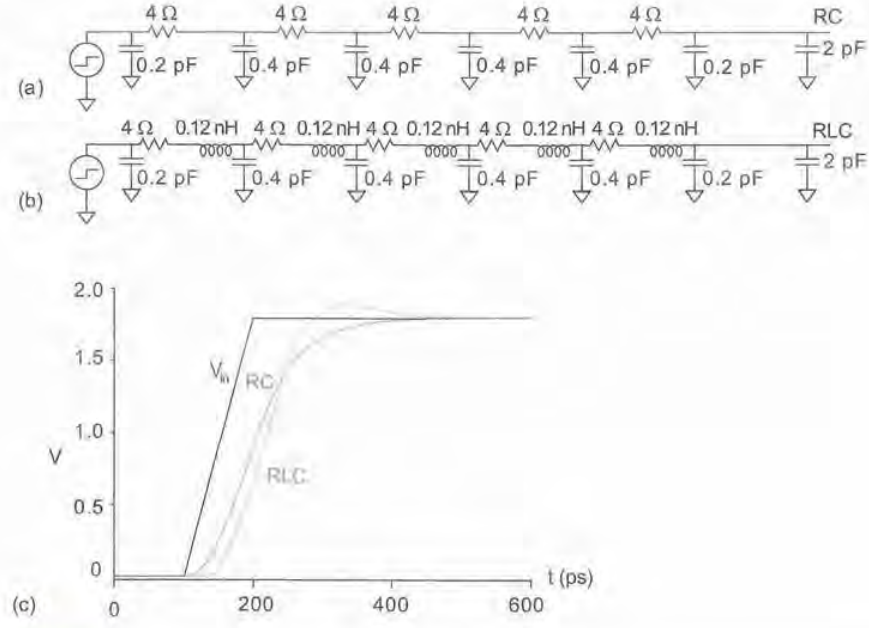


FIG 4.46 Wide clock line modeled with and without inductance

To reduce the inductance and the impact of skin effect when no ground plane is available, it is good practice to split wide wires into thinner sections interdigitated with power and ground lines to serve as return paths. For example, Figure 4.47 shows how a 16 μm wide clock line can be split into four 4 μm lines to reduce the inductance.

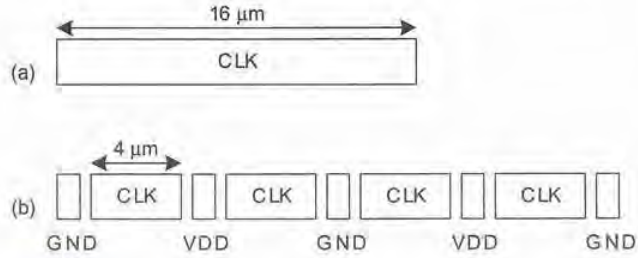


FIG 4.47 Wide clock line interdigitated with power and ground lines to reduce inductance

A bus made of closely spaced wires far above a ground plane is especially susceptible to inductive crosstalk. If all but one wire in the bus rises, each loop induces a magnetic field. These magnetic fields all pass through the loop formed by the nonswitching wire, in turn inducing a current in the victim wire. The noise from each aggressor sums on to the victim in much the same way that multiple primary turns in a transformer couple onto a single secondary turn. Computing the inductive crosstalk requires extracting a mutual inductance matrix for the bus and simulating the system. As this is not yet practical for large chips, designers instead either follow design rules that keep the inductive effects small or ignore inductance and hope for the best. The design rules may be of the form that one power or ground wire must be inserted between every N signal lines on each layer. N is called the signal:return (SR) ratio [Morton99]. $N = 4$ eliminates most inductive effects on noise and delay in a 180 nm process. $N = 2$ means each signal is shielded on one side, also eliminating half the capacitive crosstalk. However, low SR ratios are expensive in terms of metal resources.

In summary, on-chip inductance is difficult to extract. Mutual inductive coupling may occur over a long range, so inductive coupling is difficult to simulate even if accurate values are extracted. Instead, design rules are usually constructed so that inductive effects may be neglected for most structures. A regular power and ground grid or plane with no gaps is essential. Power and ground lines should be interdigitated in wide signal lines such as clocks and between about every four bits in large high-speed busses. Inductance should be incorporated into simulations of the power and clock networks and into the noise and delay calculations for busses with large SR ratios in high-speed designs.

OPTIONAL

4.5.6 Temperature Dependence

Interconnect capacitance is independent of temperature, but the resistance varies strongly. The temperature coefficients of copper and aluminum are about 0.4%/°C over the normal operating range of circuits; that is, a 100° C increase in temperature leads to 40% higher resistance. At liquid nitrogen temperature, the bulk resistivity of copper drops to 0.22 μΩ-cm, an eight-fold improvement. This suggests great advantages for RC-dominated paths in cooled systems.

OPTIONAL

4.5.7 An Aside on Effective Resistance and Elmore Delay

In this chapter we have played fast and loose with the relationship between resistance and delay. In practice, the methods work well for hand estimations. When a highly accurate result is desired, you can simulate the circuit with actual device models and wire parameters rather than simplified RC circuits. Nevertheless, rigor compels us to revisit this relationship.

According to the Elmore delay model, a gate with effective resistance R and capacitance C has a propagation delay of RC. A wire with distributed resistance R and capacitance C treated as a single π-segment has propagation delay $RC/2$. Reviewing the properties of RC circuits, we recall that the lumped RC circuit in Figure 4.48(a) has a unit step response of

$$V_{out}(t) = 1 - e^{\frac{-t}{R'C}} \tag{4.51}$$

The propagation delay of this circuit is obtained by solving for t_{pd} when $V_{out}(t_{pd}) = 1/2$:

$$t_{pd} = R'C \ln 2 = 0.69R'C \tag{4.52}$$

The distributed RC circuit in Figure 4.48(b) has no closed form time domain response. Because the capacitance is distributed along the circuit rather than all being at the end, you would expect the capacitance to be charged on average through about half the

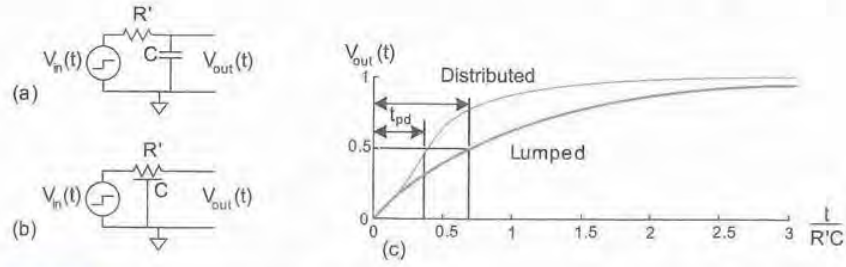


FIG 4.48 Lumped and distributed RC circuit response

resistance and that the propagation delay should thus be about half as great. A numerical analysis finds that the propagation delay is $0.38R'C$.

To reconcile the Elmore model with the true results for a logic gate, recall that logic gates have complex nonlinear I-V characteristics and are approximated as having an effective resistance. If we characterize that effective resistance as $R = R'\ln 2$, the propagation delay really becomes the product of the effective resistance and the capacitance: $t_{pd} = RC$. In Section 5.4.5, we will calculate this effective resistance by simulating the delay of a gate driving a capacitive load and measuring the propagation delay.

For distributed circuits, observe that

$$0.38R'C \approx \tfrac{1}{2}R'C\ln 2 = \tfrac{1}{2}RC\,.$$

Therefore, the Elmore delay model describes distributed delay well if we use an effective wire resistance equal to 69% of that computed with EQ (4.34). This is somewhat inconvenient. The effective resistance is further complicated by the effect of nonzero rise time on propagation delay. Figure 4.49 shows that the propagation delay depends on the rise time of the input and approaches RC for lumped systems and $RC/2$ for distributed systems when the input is a slow ramp. This suggests that when the input is slow, the effective resistance for delay calculations in a distributed RC circuit is equal to the true resistance. Finally, we note that for many analyses such as repeater insertion calculations in Section 4.6.4, the results are only weakly sensitive to wire resistance, so using the true wire resistance does not introduce great error.

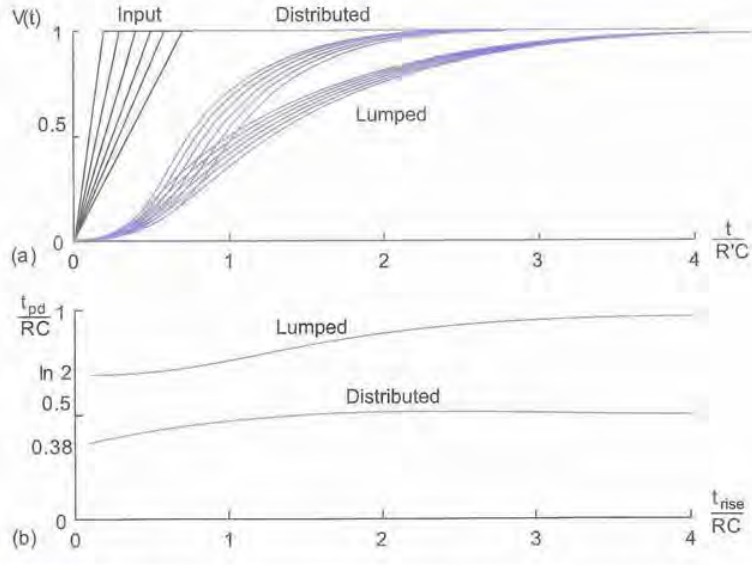


FIG 4.49 Effect of rise time on lumped and distributed RC circuit delays

In summary, it is a reasonable practice to estimate propagation delay of gates using the Elmore delay model as RC where R is the effective resistance of the gate. Similarly, you can estimate the flight time along a wire as $RC/2$ where R is the true resistance of the wire. When more accurate results are needed, it is important to use good transistor models and appropriate input slopes in simulation.

On a separate topic, Section 4.2.1 defined the Elmore delay of an RC ladder. In general, a wire branching to many destinations is usually modeled as an *RC tree* instead. The Elmore delay to node i of an RC tree is

$$T_{D_i} = \sum_{k=1}^{N} R_{ki} C_k \tag{4.53}$$

where N is the number of nodes in the tree, C_k is the capacitance on node k, and R_{ki} is the resistance between the input and node k in common with the path between the input and node i. This simplifies to the simple product of resistance and capacitance for each node in an RC ladder. In trees, the capacitance on branches away from the path to the output is conservatively lumped as if it were at the branch point on the path.

Example

Figure 4.50 models a gate driving wires to two destinations. The gate is represented as a voltage source with effective resistance R_1. The two receivers are located at nodes 3 and 4. The wire to node 3 is long enough that it is represented with a pair of π-segments, while the wire to node 4 is represented with a single segment. Find the Elmore delay from input x to each receiver.

Solution: The Elmore delays are:

$$\begin{aligned} T_{D_3} &= R_1C_1 + (R_1 + R_2)C_2 + (R_1 + R_2 + R_3)C_3 + R_1C_4 \\ T_{D_4} &= R_1C_1 + R_1C_2 + R_1C_3 + (R_1 + R_4)C_4 \end{aligned} \tag{4.54}$$

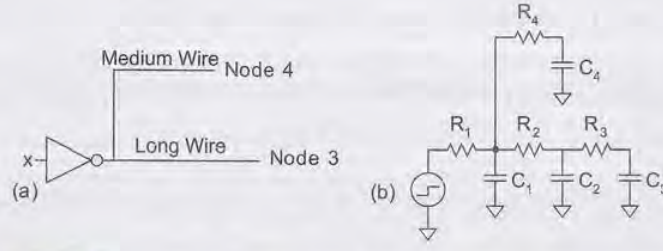


FIG 4.50 Interconnect modeling with RC tree

The Elmore delay can be viewed in terms of the first moment of the impulse response of the circuit. CAD tools can obtain greater accuracy by approximating delay based on higher moments using a technique called *moment matching*. *Asymptotic Waveform Evaluation* (AWE) uses moment matching to estimate interconnect delay with better accuracy than the Elmore delay model and faster run times than a full circuit simulation [Celik02].

4.6 Wire Engineering

As gate delays continue to improve while long wire delays remain constant or even get slower, wire engineering has become a major part of integrated circuit design. It is necessary to develop a floorplan early in the design cycle, identify the long wires, and plan for them. While floorplanning in such a way that critical communicating units are close to one another has the greatest impact on performance, it is inevitable that long wires will still exist. The designer has a number of techniques to engineer wires for delay and coupling noise. The width, spacing, and layer usage are all under the designer's control. Shielding can be used to further reduce coupling on critical nets. Repeaters inserted along long wires reduce the delay from a quadratic to a linear function of length. Wire capacitance and resistance complicate the use of Logical Effort in selecting gate sizes.

4.6.1 Width and Spacing

The designer selects the wire width, spacing, and layer usage to achieve acceptable delay and noise. By default, minimum pitch wires are preferred for noncritical interconnections for best density. Widening a wire proportionally reduces resistance but increases the capacitance of its top and bottom plates. Table 4.8 showed that this leads to less than a proportional increase in capacitance, so the RC delay product improves, especially for narrow wires. Widening wires also increases the fraction of capacitance of the top and bottom plates, which somewhat reduces coupling noise from adjacent wires. Increasing spacing between wires reduces capacitance to the adjacent wires and leaves resistance unchanged. This improves the RC delay to some extent and significantly reduces coupling noise.

Figure 4.51(a) shows the RC delay of a 10 mm metal2 wire in the process from Section 4.5 sandwiched between metal1 and metal3 planes as a function of wire pitch ($w + s$) for different wire spacings. Figure 4.51(b) shows the coupling capacitance to the two adjacent neighbors as a fraction of the total capacitance. The data shows that for tight pitches, it is better to increase width than spacing to improve delay. However, it is better to increase spacing than width to improve coupling. The best tradeoff clearly depends on the situation.

4.6.2 Layer Selection

Early (1970s) MOS processes offered only a single layer of metal. Polysilicon or diffusion jumpers were required when two signals crossed. Modern processes have six or more metal layers. The lower layers are thin and optimized for a tight routing pitch. Middle layers are

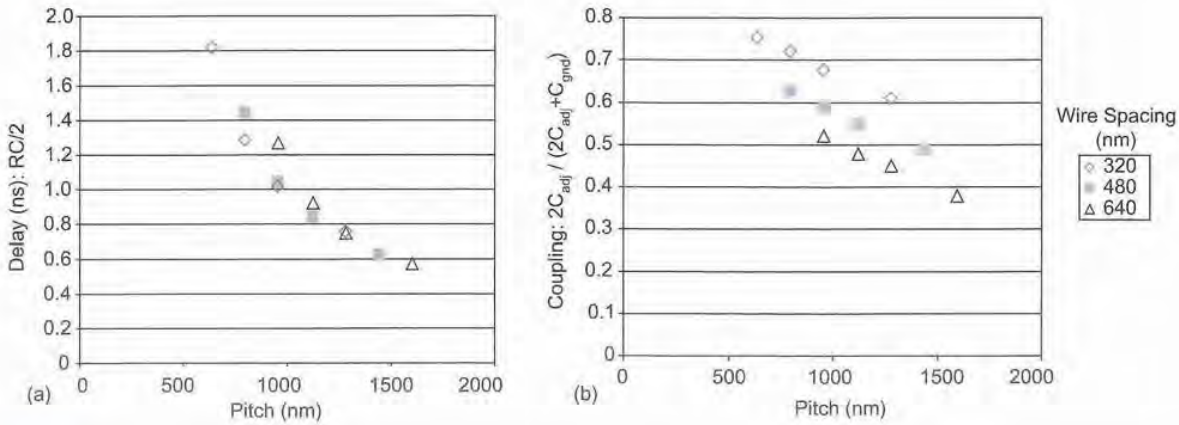


FIG 4.51 Delay and coupling of 10 mm metal2 wire for various wire pitches and spacings

often slightly thicker for lower resistance and better current-handling capability. Upper layers may be even thicker to provide a low-resistance power grid and fast global interconnect. A large number of layers are important to be able to route complex chips and supply power, ground, and the clock. Additional layers are also valuable because they allow a relaxed wire pitch, reducing RC delay and coupling problems. Wiring tracks are a precious resource and are often allocated in the floorplan; the wise designer maintains a reserve of wiring tracks for unanticipated changes late in the design process. A sample allocation of wiring tracks for a six-layer metal process is given in Table 4.11. Critical signals within a unit can be assigned upper-level metal tracks to reduce delay.

Table 4.11 Sample metal layer usage in 6-level process

Layer	Purpose
Metal 1	Interconnect within cells
Metal 2/3	Interconnect between cells within units
Metal 4/5	Interconnect between units, critical signals
Metal 6	I/O pads, clock, power, ground

The power grid is usually distributed over multiple layers. Most of the current-handling capability is provided in the upper two layers with lowest resistance. However, the grid must extend down to metal1 or metal2 to provide easy connection to cells.

There is debate over the best use of the growing number of wiring layers to solve inductive and capacitive noise problems. One approach is to dedicate power and ground

planes, much like in a printed circuit board. For example, the Alpha 21264 [Gronowski98] used thick metal3 and metal6 planes for ground and power. This was necessary because the I/O pins were all along the periphery of the chip and the planes were needed to provide a low-resistance path for power to the center; better packaging with power and ground bumps across the die reduces the pressure to use planes. Another approach is to shield noise-sensitive nets with power or ground lines; this will be discussed in Section 4.6.3. A third approach is to use differential signaling.

4.6.3 Shielding

As discussed in Section 4.5.4, coupling from adjacent lines impacts both the delay and signal integrity of wires. The coupling can be avoided if the adjacent lines do not switch. It is common practice to shield critical signals with power or ground wires on one or both sides to eliminate coupling. This is costly in area but may be less costly than increasing spacing to the point that the coupling is negligible. For example, clock wires are usually shielded so that switching neighbors do not affect the delay of the clock wire and introduce clock skew. Sensitive analog wires passing near digital signals should also be shielded.

An alternative to shielding is to interdigitate wires that are guaranteed to switch at different times. For example, if bus *A* switches on the rising edge of the clock and bus *B* switches on the falling edge of the clock, by interleaving the bits of the two busses you can guarantee that both neighbors are constant during a switching event. This avoids the delay impact of coupling; however, you must still ensure that coupling noise does not exceed noise budgets. Figure 4.52 shows wires shielded (a) on one side, (b) on both sides, and (c) interdigitated. Very sensitive signals such as clocks or analog voltages can be shielded above and below as well.

4.6.4 Repeaters

Both resistance and capacitance increase with wire length l, so the RC delay of a wire increases with l^2, as shown in Figure 4.53(a). The delay may be reduced by splitting the wire into N segments and inserting an inverter or buffer called a *repeater* to actively drive the wire [Glasser85], as shown in Figure 4.53(b). The new wire involves N segments with RC flight time of $(l/N)^2$, for a total delay of l^2/N. If the number of segments is proportional to the length, the overall delay increases only linearly with l.

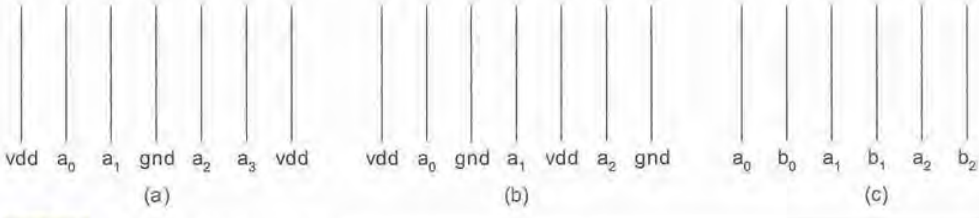


FIG 4.52 Wire shielding topologies

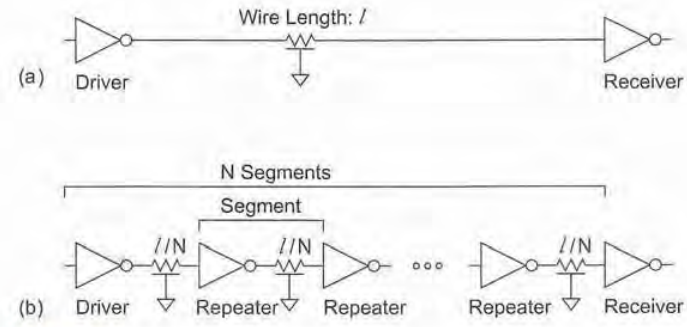


FIG 4.53 Wire with and without repeaters

Using inverters as repeaters gives best performance. Each repeater adds some delay. If the distance is too great between repeaters, the delay will be dominated by the long wires. If the distance is too small, the delay will be dominated by the large number of inverters. As usual, the best distance between repeaters is a compromise between these extremes. Suppose a unit inverter has resistance R and capacitance C' ($C' \approx 3C$ because the inverter is composed of a unit nMOS and double-width pMOS[8]) and a wire has resistance R_w and capacitance C_w per unit length. Consider inserting repeaters of W times unit size. You can show (see Exercise 33) that under the Elmore delay model, neglecting diffusion parasitics, the best length of wire between repeaters is

$$\frac{l}{N} = \sqrt{\frac{2RC'}{R_w C_w}} \tag{4.55}$$

The delay per unit length of a properly repeated wire is

$$\frac{t_{pd}}{l} = \left(2+\sqrt{2}\right)\sqrt{RC'R_w C_w} \tag{4.56}$$

To achieve this delay, the inverters should use an nMOS transistor width of

$$W = \sqrt{\frac{RC_w}{R_w C'}} \tag{4.57}$$

Unfortunately, inverting repeaters complicate design because you must either ensure an even number of repeaters on each wire or adapt the receiving logic to accept an inverted

[8]It is not necessary to use equal rise and fall resistances and Section 6.2.1.6 shows that the delay and area can be slightly improved using smaller pMOS transistors.

input. Some designers use inverter pairs (buffers) rather than single inverters to avoid the polarity problem. The pairs contribute more delay. However, the first inverter size W_1 may be smaller, presenting less load on the wire driving it. The second inverter may be larger, driving the next wire more strongly. You can show that the best size of the second inverter is $W_2 = kW_1$, where $k = 2.06$ if diffusion parasitics are negligible. The distance between repeaters increases to (see Exercise 34)

$$\frac{l}{N} = \sqrt{\frac{2RC'\left(k + \frac{1}{k}\right)}{R_w C_w}} \tag{4.58}$$

The delay per unit length becomes

$$\frac{t_{pd}}{l} = 3.65\sqrt{RC'R_w C_w} \tag{4.59}$$

using transistor widths of

$$W_1 = \frac{W}{\sqrt{k}}, \quad W_2 = W\sqrt{k} \tag{4.60}$$

This typically means that wires driven with noninverting repeaters are only about 7% slower per unit length than those using inverting repeaters. Only about two-thirds as many repeaters are required, simplifying floorplanning. Total repeater area and power increases slightly.

When diffusion parasitics are considered, each repeater becomes slightly more expensive. Therefore, it is better to use fewer repeaters spaced at greater distances, as you would expect from Logical Effort. The best transistor sizes are unchanged. There are no closed-form results, but the problem can be solved numerically or thorough simulation. The overall delay is a weak function of the distance between repeaters, so it is reasonable to increase this distance to reduce the difficulty of finding places in the floorplan for repeaters while only slightly increasing delay. Repeaters impose directionality on a wire. Bidirectional busses and distributed tristate busses cannot use simple repeaters and hence are slower; this favors point-to-point unidirectional communications. Repeaters also draw unusually large crowbar currents because their inputs come from RC lines with slow edge rates, turning both transistors partially ON for a long time.

An alternative to repeaters are *boosters* [Nalamalpu02], which are placed in parallel rather than in series with a wire as shown in Figure 4.54(a). The booster senses when a wire is switching and aids the transition. It relies on the principles of hysteresis and positive feedback to allow bidirectional operation at the expense of reduced noise margins. The

Example

Determine the best distance between repeaters for a minimum pitch metal2 line in a 180 nm process. Assume the metal usage on other layers is dense. The transistor resistance is 3kΩ • μm and the gate capacitance is C = 1.7 fF/μm. How far should the repeaters be spaced? How wide should the repeater transistors be? What is the signal velocity on the wire? How do your results change if width and spacing are each increased by 50%? How do they change for a minimum-pitch metal5 line?

Solution: The capacitance of minimum-pitch metal2 lines with planes above and below is C_w = 0.21 fF/μm, according to Table 4.8. The sheet resistance from Table 4.7 is 0.05 Ω/square, or

$$R_w = \frac{0.05 \frac{\Omega}{\text{square}}}{0.32 \frac{\mu m}{\text{square}}} = 0.16 \frac{\Omega}{\mu m} \tag{4.61}$$

$C' = 3C$ = 5.1 fF/μm. According to EQ (4.55), the distance between repeaters should be

$$\sqrt{\frac{2(3000\Omega \cdot \mu m)\left(5.1 \frac{fF}{\mu m}\right)}{\left(0.16 \frac{\Omega}{\mu m}\right)\left(0.21 \frac{fF}{\mu m}\right)}} = 950\ \mu m \tag{4.62}$$

According to EQ (4.57), the nMOS transistor width should be

$$\sqrt{\frac{(3000\Omega \cdot \mu m)\left(0.21 \frac{fF}{\mu m}\right)}{\left(0.16 \frac{\Omega}{\mu m}\right)\left(5.1 \frac{fF}{\mu m}\right)}} = 28\ \mu m \tag{4.63}$$

and the pMOS should be roughly twice that. The repeated signal velocity (ignoring diffusion capacitance) is

$$\left(2+\sqrt{2}\right)\sqrt{(3000\Omega \cdot \mu m)\left(5.1 \frac{fF}{\mu m}\right)\left(0.16 \frac{\Omega}{\mu m}\right)\left(0.21 \frac{fF}{\mu m}\right)} = 77 \frac{ps}{mm} \tag{4.64}$$

In comparison, the speed of light in a medium with a relative permittivity of 3.55 is

$$\frac{3 \cdot 10^8 \frac{\text{m}}{\text{s}}}{\sqrt{3.55}} = 0.155 \frac{\text{mm}}{\text{ps}} = 6.3 \frac{\text{ps}}{\text{mm}} \tag{4.65}$$

Even with repeaters, the wire is an order of magnitude slower than the speed of light because it is so resistive.

If the width and spacing increase by 50%, the capacitance is 0.18 fF/μm and the resistance is

$$R_w = \frac{0.05 \frac{\Omega}{\text{square}}}{1.5 \cdot 0.32 \frac{\mu\text{m}}{\text{square}}} = 0.10 \frac{\Omega}{\mu\text{m}} \tag{4.66}$$

As the wire is faster, we would expect that the repeaters could be placed further apart but that larger transistors would be needed to drive the longer segments. Reevaluating the same equations gives a distance of 1300 μm between repeaters, transistor widths of 33 μm, and wire delay of 52 ps/mm.

A metal5 line is even faster with a capacitance of 0.24 fF/μm and resistance of

$$R_w = \frac{0.02 \frac{\Omega}{\text{square}}}{0.8 \frac{\mu\text{m}}{\text{square}}} = 0.025 \frac{\Omega}{\mu\text{m}} \tag{4.67}$$

The best repeater spacing is 2260 μm with 75 μm transistors, giving a delay of 30 ps/mm. It is clear that the fat, wide upper level metal lines are a precious routing resource.

gate in the center is an *inverting Muller C-element*, defined in Figure 4.54(b), which provides hysteresis by only toggling when both inputs are the same. Skewed inverters with high and low switching points sense the beginning of a transition. Figure 4.54(c) shows simulations of the booster in action on a 6mm long metal2 wire. The wire is initially '0,' so *l* and *h* are initially '1' and *c* is initially '0.' In this state, the booster is OFF and the weak

keeper holds the wire low. The input rises and the middle of the wire mid slowly follows. The keeper opposes the transition but is weak enough to have little effect. When the middle reaches the switching point of the LO-skew inverter, node *l* falls. Now the booster fires through both pMOS transistors, serving as positive feedback to strongly pull the middle of the wire high. This is visible as a kink in the mid waveform. The keeper also turns OFF. As the middle reaches the switching point of the HI-skew inverter, node *h* falls. *c* rises and turns OFF the booster because the transition is nearly complete. The keeper turns ON again, holding the middle of the wire high. The waveforms without boosters are also shown; they are slowly rising exponentials. Nalamalpu finds boosters are 20% faster and are inserted at three times the spacing of repeaters. [Dobbalaere95] presents a similar technique using self-timed circuits.

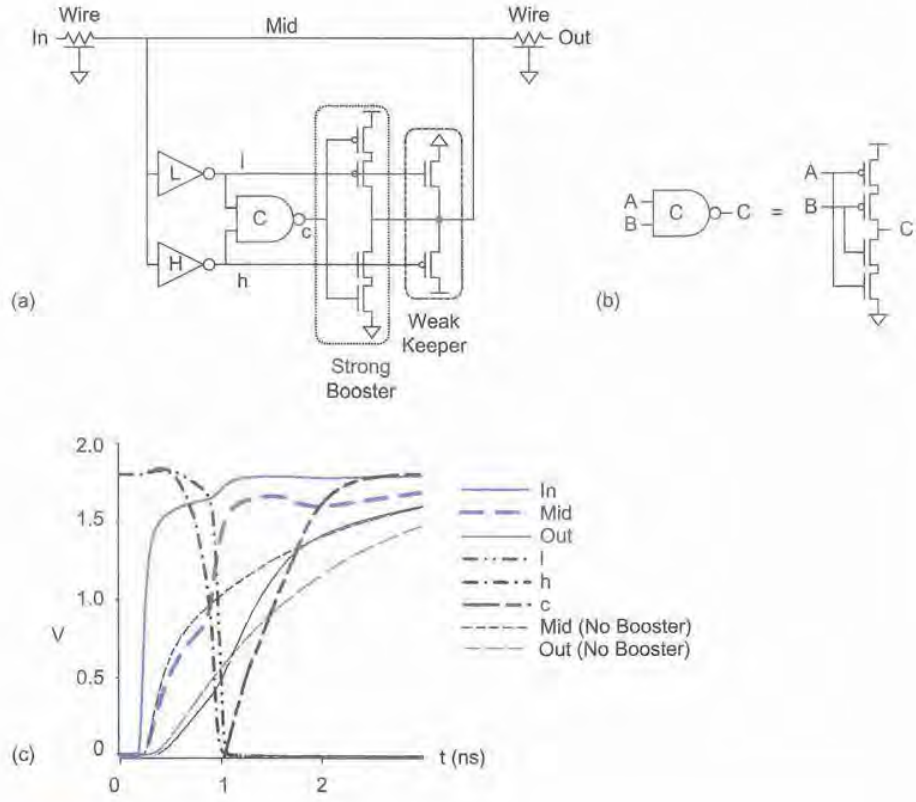


FIG 4.54 Booster

4.6.5 Implications for Logical Effort

Interconnect complicates the application of Logical Effort because the wires have a fixed capacitance. The branching effort at a wire with capacitance C_{wire} driving a gate load of C_{gate} is $(C_{gate} + C_{wire}) / C_{gate}$. This branching effort is not constant; it depends on the size of the gate being driven. The simple rule that circuits are fastest when all stages bear equal effort is no longer true when wire capacitance is introduced. If the wire is very short or very long, there are simple approximations possible, but when the wire and gate loads are comparable, there is no simple method to determine the best stage effort.

Every circuit has some interconnect, but when the interconnect is short ($C_{wire} << C_{gate}$), it can be ignored. Alternatively, you can compute the average ratio of wire capacitance to parasitic diffusion capacitance and add this as extra parasitic capacitance when determining parasitic delay. For connections between nearby gates, this generally leads to a best stage effort ρ slightly greater than 4.

Conversely, when the interconnect is long ($C_{wire} >> C_{gate}$), the gate at the end can be ignored. The path can now be partitioned into two parts. The first part drives the wire while the second receives its input from the wire. The first part is designed to drive the load capacitance of the wire; the extra load of the receiver is negligible. A long wire is often driven by an inverter with a stage effort of 8 to 12 rather than 4 because low stage efforts require large, power-hungry drivers and have little performance advantage when the delay is dominated by the wire RC. The size of the receiver is chosen by practical considerations: Larger receivers may be faster, but they also cost area and power.

The most difficult problems occur when $C_{wire} \approx C_{gate}$. These medium-length wires introduce branching efforts that are a strong function of the size of the gates they drive. Writing a delay equation as a function of the gate sizes along the path and the wire capacitance results in an expression that can be differentiated with respect to gate sizes to compute the best sizes. For the purpose of hand estimation, this is usually too much work. A reasonable alternative is to preserve the stage effort of about four. The initial branching effort of the wire is unknown, so iteration is usually necessary.

4.6.6 Crosstalk Control



Recall from EQ (4.40) that the capacitive crosstalk is proportional to the ratio of coupling capacitance to total capacitance. For modern wires with an aspect ratio (t/w) of 2 or greater, the coupling capacitance can account for two-thirds or more of the total capacitance and crosstalk can create large amounts of noise and huge data-dependent delay variations. Shielding or increasing the spacing or width of wires alleviates crosstalk at the expense of greater area. Three alternative techniques to control crosstalk are *staggered repeaters*, *charge compensation*, and *twisted differential signaling* [Ho03b]. Each technique seeks to cause equal amounts of positive and negative crosstalk on the victim, effectively producing zero net crosstalk.

Figure 4.56(a) shows two wires with staggered repeaters. Each segment of the victim sees half of a rising aggressor segment and half of a falling aggressor segment. Although the cancellation is not perfect because of delays along the segments, staggering is a very

Example

The path in Figure 4.55 contains a medium-length wire modeled as a lumped capacitance. Write an equation for path delay in terms of x and y. How large should the x and y inverters be for shortest path delay? What is the stage effort of each stage?

Solution: From the Logical Effort delay model, we find the path delay is

$$d = \frac{x}{10} + \frac{y+50}{x} + \frac{100}{y} + P \tag{4.68}$$

Differentiating with respect to each size and setting the results to 0 allows us to solve EQ (4.68) for x = 33 fF and y = 57 fF. The stage efforts are (33/10) = 3.3, (57 + 50)/33 = 3.2, and (100/57) = 1.8. Notice that the first two stage efforts are equal as usual, but the third stage effort is lower. As x already drives a large wire capacitance, y may be rather large (and will bear a small stage effort) before the incremental increase in delay of x driving y equals the incremental decreases in delay of y driving the output.

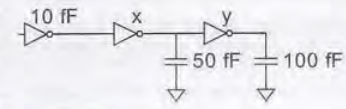


FIG 4.55 Path with medium-length wire

$$\frac{1}{10} - \frac{y+50}{x^2} = 0 \Rightarrow x^2 = 10y + 500$$
$$\frac{1}{x} - \frac{100}{y^2} = 0 \Rightarrow y^2 = 100x \tag{4.69}$$

effective approach. Figure 4.56(b) shows charge compensation in which an inverter and transistor are added between the aggressor and victim. The transistor is connected to behave as a capacitor. When the aggressor rises and couples the victim upward, the inverter falls and couples the victim downward. By choosing an appropriately sized compensation transistor, most of the noise can be cancelled at the expense of the extra circuitry. Figure 4.56(c) shows twisted differential signaling in which each signal is routed differentially. The signals are swapped or *twisted* such that the victim and its complement each see equal coupling from the aggressor and its complement. This approach is expensive in wiring resources, but it very effectively eliminates crosstalk. It is widely used in memory designs, as explored in Section 11.2.3.

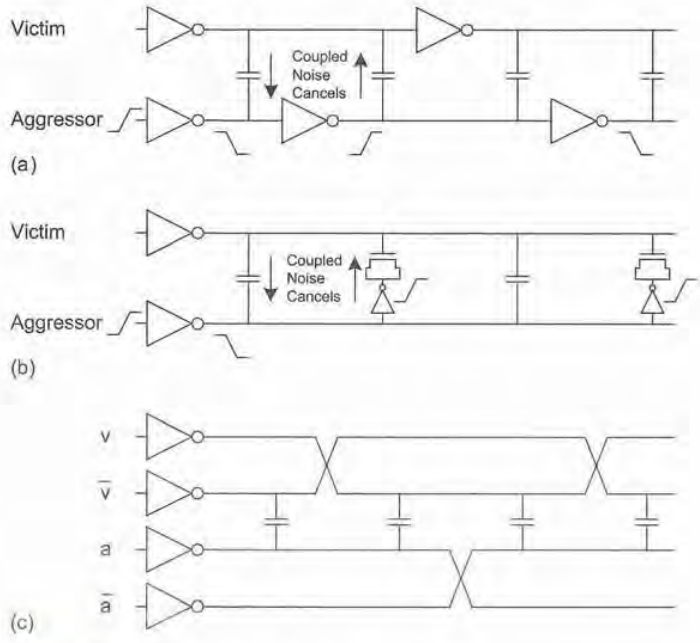


FIG 4.56 Crosstalk control schemes

4.6.7 Low-swing Signaling

OPTIONAL

Driving long wires is slow because of the RC delay, and expensive in power because of the large capacitance to switch. Low-swing signaling improves performance by sensing when a wire has swung through some small V_{swing} rather than waiting for a full swing. If the driver is turned off after the output has swung sufficiently, the power can be reduced as well. However, the improvements come at the expense of more complicated driver and receiver circuits. Low-swing signaling may also require a twisted differential pair of wires to eliminate common-mode noise that could corrupt the small signal.

The power consumption for low-swing signaling depends on both the driver voltage V_{drive} and the actual voltage swing V_{swing}. Each time the wire is charged and discharged, it consumes $Q = CV_{swing}$. If the effective switching frequency of the wire is αf, the average current is

$$I_{avg} = \frac{1}{T}\int_0^T i_{drive}(t) = \alpha f C V_{swing} \tag{4.70}$$

Hence, the dynamic dissipation is

$$P_{\text{dynamic}} = I_{\text{avg}} V_{\text{drive}} = \alpha f C V_{\text{swing}} V_{\text{drive}} \tag{4.71}$$

In contrast, a rail-to-rail driver uses $V_{\text{drive}} = V_{\text{swing}} = V_{DD}$ and thus consumes power proportional to V_{DD}^2 (EQ (4.30)). V_{swing} must be less than or equal to V_{drive}. By making V_{swing} less than V_{drive}, we speed up the wire because we do not need to wait for a full swing. By making both voltages significantly less than V_{DD}, we can reduce the power by an order of magnitude.

Low-swing signaling involves numerous challenges. A low V_{drive} must be provided to the chip and distributed to low-swing drivers. The signal should be transmitted on differential pairs of wires that are twisted to cancel coupling from neighbors and equalized to prevent interference from the previous data transmitted. The driver must turn on long enough to produce V_{swing} at the far end of the line, then turned off to prevent unnecessary power dissipation. This generally leads to a somewhat larger swing at the near end of the line. The receiver must be clocked at the appropriate time to amplify the differential signal. Distributing a self-timed clock from driver to receiver is difficult because the distances are long so the time to transmit a full-swing clock exceeds the time for the data to complete its small swing.

[Ho03a] describes a synchronous low-swing signaling technique using the system clock for both driver and receiver. During the first half of the cycle, the driver is OFF (high impedance) and the differential wires are equalized to the same voltage. During the second half of the cycle, the drivers turn ON. At the end of the cycle, the receiver senses the differential voltage and amplifies it to full-swing levels. Figure 4.57(a) shows the overall system architecture. Figure 4.57(b) shows the driver for one of the wires. The gates use ordinary V_{DD} while the drive transistors use V_{drive}. Because $V_{\text{drive}} << V_{DD}$, nMOS transistors are used for both the pullup and pulldown to deliver low effective resistance in their linear regime. A second driver using the complementary input drives the complementary wire. Figure 4.57(c) shows the differential wires with twisting and equalizing. Figure 4.57(d) shows the clocked sense amplifier based on the SA-F/F that will be described further in Section 7.3.8. The sense amplifier uses pMOS input transistors because the small-swing inputs are close to GND and below the threshold of nMOS transistors.

The design in [Ho03a] used a 1.8 V V_{DD}, 0.65 V V_{drive}, and 0.1 V minimum V_{swing} in a 180 nm process. The clock frequency was 1 GHz, or 10 FO4 inverter delays, to drive a 10 mm wire. Note that the clock period must be long enough to transmit an adequate voltage swing. If the clock period increases, the circuit will actually dissipate more power because the voltage swing will increase to a maximum of V_{drive}.

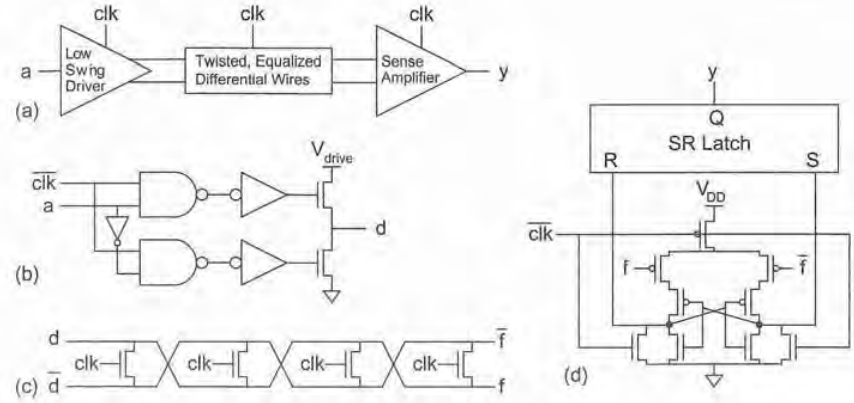


FIG 4.57 Low-swing signaling system

4.7 Design Margin

So far, when considering the various aspects of determining a circuit's behavior, we have only alluded to the variations that might occur in this behavior given different operating conditions. In general, there are three different sources of variation—two environmental and one manufacturing. These are:

- Supply voltage
- Operating temperature
- Process variation

You must aim to design a circuit that will reliably operate over all extremes of these three variables. Failure to do so invites circuit failure, potentially catastrophic system failure, and a rapid decline in reliability (not to mention a loss of customers).

Variations can be modeled with *uniform* or *normal* (Gaussian) statistical distributions. These distributions are shown in Figure 4.58. Uniform distributions are specified with a half-range. For good results, accept variations over the entire half-range. For example, V_{DD} may be specified at 1.2 V +/− 10%. This is a uniform distribution with a 120 mV half-range and all parts should work at any voltage in the range. Normal distributions are specified with a standard deviation σ. Processing variations are usually modeled with normal distributions. Retaining parts with a 3σ distribution will result in 0.26% of parts being rejected. A 2σ retention results in 4.56% of parts being rejected, while 1σ results in a 31.74% rejection rate. Obviously, rejecting parts outside 1σ of nominal would waste a large number of parts. A 3σ or 2σ limit is conventional and a manufacturer with a com-

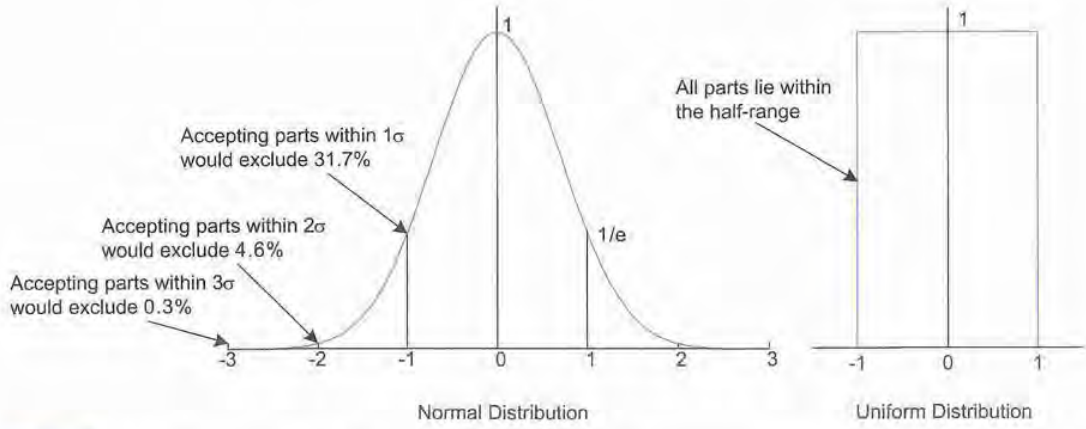


FIG 4.58 Uniform and normal distributions

mercially viable CMOS process should be able to supply a set of device parameters that are guaranteed to yield at this rate. However, as the variations are getting more significant, designers are moving toward statistical rather than worst-case design [Borkar03].

4.7.1 Supply Voltage

Systems are designed to operate at a nominal supply voltage, but this voltage may vary for many reasons including tolerances of the voltage regulator, IR drops along supply rails, and di/dt noise. The system designer may tradeoff power supply noise against resources devoted to power supply regulation and distribution; typically the supply is specified at ±10% around nominal at each logic gate. In other words, the variation has a uniform distribution with a half-range of 10% of V_{DD}. Speed is roughly proportional to V_{DD}, so to first order this leads to ±10% delay variations (check for your process and voltage when this is critical). Power supply variations also appear in noise budgets.

4.7.2 Temperature

Section 2.2.3.7 showed that as temperature increases, drain current decreases. The junction temperature of a transistor is the sum of the ambient temperature and the temperature rise caused by power dissipation in the package. This rise is determined by the power consumption and the package thermal resistance, as discussed in Section 12.2.4.

Table 4.12 lists the ambient temperature ranges for parts specified to commercial, industrial, and military standards. Parts must operate at the bottom end of the ambient range unless they are allowed time to warm up before use. The junction temperature may significantly exceed the maximum ambient temperature. Commonly commercial parts are verified to operate with junction temperatures up to 110° to 125° C.

Table 4.12 Ambient temperature ranges

Standard	Minimum	Maximum
Commercial	0° C	70° C
Industrial	−40° C	85° C
Military	−55° C	125° C

4.7.3 Process Variation

Devices and interconnect have variations in film thickness, lateral dimensions, and doping concentrations [Bernstein99]. These variations occur from one wafer to another, between dice on the same wafer, and across an individual die; variation is generally smaller across a die than between wafers. These effects are sometimes called *inter-die* and *intra-die* variations; intra-die variation is also called *process tilt* because certain parameters may slowly and systematically vary across a die. For example, if an ion implanter delivered a greater dose nearer the center of a wafer than near the periphery, the threshold voltages might tilt radially across the wafer.

For devices, the most important variations are channel length L, oxide thickness t_{ox}, and threshold voltage V_t. Channel length variations are caused by photolithography proximity effects, deviations in the optics, and plasma etch dependencies. Oxide thickness is well controlled and generally is only significant between wafers; its effects on performance are often lumped into the channel length variation. Threshold voltages vary because of different doping concentrations and annealing effects, mobile charge in the gate oxide, and discrete dopant variations caused by the small number of dopant atoms in tiny transistors.

For interconnect, the most important variations are line width and spacing, metal and dielectric thickness, and contact resistance. Line width and spacing, like channel length, depend on photolithography and etching proximity effects. Thickness may be influenced by polishing. Contact resistance depends on contact dimensions and the etch and clean steps.

4.7.4 Design Corners

From the designer's point of view, the collective effects of process and environmental variation can be lumped into their effect on transistors: *typical* (also called *nominal*), *fast*, or *slow*. In CMOS, there are two types of transistors with somewhat independent characteristics, so the speed of each can be characterized. Moreover, interconnect speed may vary independently of devices. When these processing variations are combined with the environmental variations, we define *design* or *process corners*. The term *corner* refers to an imaginary box that surrounds the guaranteed performance of the circuits, shown in Figure 4.59. The box is not square because some characteristics such as oxide thickness track between devices, making it impossible to simultaneously find a slow nMOS transistor with thick oxide and a fast pMOS transistor with thin oxide.

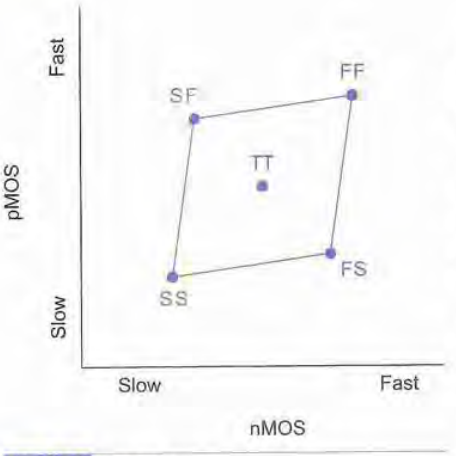


FIG 4.59 Design corners

Table 4.13 lists a number of important design corners. The corners are specified with five letters describing the nMOS, pMOS, interconnect, power supply, and temperature, respectively. The letters are F, T, and S, for *fast*, *typical*, and *slow*. The environmental corners for a 1.8 V commercial process are shown in Table 4.14, illustrating that circuits are fastest at high voltage and low temperature. Circuits are most likely to fail at the corners of the design space, so nonstandard circuits should be simulated at all corners to ensure they operate correctly in all cases. Often integrated circuits are designed to meet a timing specification for typical processing. These parts may be *binned*; faster parts are rated for higher frequency and sold for more money, while slower parts are rated for lower frequency. In any event, the parts must still work in the slowest SSSSS environment. Other integrated circuits are designed to obtain high yield at a relatively low frequency; these parts are simulated for timing in the slow process corner. The fast corner FFFFF has maximum DC power dissipation because threshold voltages are lowest. Other corners are used to check for races and ratio problems where the relative strengths and speeds of different transistors or interconnect are important. The FFFFS corner is important for noise because the edge rates are fast, causing more coupling; the threshold voltages are low; and the leakage is high [Shepard99].

Often the corners are abbreviated to fewer letters. For example, two letters generally refer to nMOS and pMOS. Three refer to nMOS, pMOS, and environment. Four refer to nMOS, pMOS, voltage, and temperature.

Table 4.13 Design corner checks

Corner					Purpose
nMOS	pMOS	Wire	V_{DD}	Temp	
T	T	T	S	S	timing specifications (binned parts)
S	S	S	S	S	timing specifications (conservative)
F	F	F	F	F	DC power dissipation, race conditions, hold time constraints, pulse collapse, noise
F	F	F	F	S	subthreshold leakage noise, overall noise analysis
S	S	F	S	S	races of gates against wires
F	F	S	F	F	races of wires against gates
S	F	T	F	F	pseudo-nMOS & ratioed circuits noise margins, memory read/write, race of pMOS against nMOS
F	S	T	F	F	ratioed circuits, memory read/write, race of nMOS against pMOS

Table 4.14 Environmental corners

Corner	Voltage	Temperature
F	1.98	0° C
T	1.8	70° C
S	1.62	125° C

It is important to know the design corner when interpreting delay specifications. For example, the datasheet in Figure 4.25 is specified at the 25° C TTTT corner. The SS corner is 27% slower. The cells are derated at −71% per volt and 0.13%/°C, for additional penalties of 13% each in the low voltage and high temperature corners. These factors are multiplicative, giving SSSS delay of 1.62 times nominal.

[Ho01] and [Chinnery02] find the FO4 inverter delay can be estimated from the effective channel length L_{eff} as:

- L_{eff} • (0.36 ps/nm) in TTTT corner
- L_{eff} • (0.50 ps/nm) in TTSS corner
- L_{eff} • (0.60 ps/nm) in SSSS corner

Note that the effective channel length is aggressively scaled faster than the drawn channel length to improve performance. Typically L_{eff} = 0.5-0.7 L_{drawn}. For example, Intel's 180 nm process was originally manufactured with L_{eff} = 140 nm and eventually pushed to L_{eff} = 100 nm. Our model predicts an FO4 inverter delay of about 70–50 ps in the TTSS corner where design usually takes place.

In addition to working at the standard process corners, chips must function in a very high temperature, high voltage *burn-in* corner (e.g., 125–140° C externally, corresponding to an even higher internal temperature, and 1.3–1.7x nominal V_{DD} [Vollertsen99]), as described in Section 9. While it does not have to run at full speed, it must operate correctly so that all nodes can toggle. The burn-in corner has very high leakage and can dictate the size of keepers and weak feedback on domino gates and static latches.

Processes with multiple threshold voltages and/or multiple oxide thicknesses can see each flavor of transistor independently varying as fast, typical, or slow. This can easily lead to more corners than anyone would care to simulate and raises challenges about identifying what corners must be checked for different types of circuits.

4.7.5 Matching



There are many cases in which the designer cares how well two nominally identical transistors match. For example, in a sense amplifier, the minimum voltage that reliably can be sensed depends on the offset voltage of the amplifier, which in turn depends on mismatch between the input transistors. Differential pairs used in most analog circuits are also very sensitive to mismatch between input transistors. In a clock distribution network, we would like to distribute a clock to all points on a chip at the same time and mismatch leads to

clock skew. It is clearly overkill to model one transistor as fast and an adjacent transistor as slow because nearby transistors experience similar processing; such conservative modeling would make design impossible. On the other hand, it is also clear that the two transistors are not identical. Designers should be able to obtain data about process variation from the manufacturer. Unfortunately, this data is often not available. Even if the manufacturer has characterized the process for variation, the data is usually a closely guarded trade secret. Moreover, manufacturers do not like to share the information for fear that it will become part of the process specification and prevent process modifications that lead to faster transistors at the expense of different variabilities. Nevertheless, device matching is a critical parameter for calculating clock skew and is an increasingly important specification for a process.

Mismatches occur from both *systematic variability* and *uncertainty* [Naffziger02]. Systematic variability has a quantitative relationship to a source. For example, an ion implanter can systematically deliver a different dosage to different regions of a wafer. Similarly, polysilicon gates can systematically be etched narrower in regions of high polysilicon density than low density. Uncertainty occurs when the source is unknown, random, or too costly to model. Systematic variability can be modeled and nulled out; for example, in principle, you could examine a layout database and calculate the etching variations as a function of nearby layout, then simulate a circuit with suitably adjusted gate lengths. In practice, systematic variability often must be treated as uncertainty because of limited modeling resources. *Process tilt* describes the tendency for parameters to slowly vary from one corner of the die to another. Adjacent transistors tend to match better than widely separated ones.

Variations in transistor threshold voltage and current have been found experimentally to scale with $1/\sqrt{WL}$ [Pelgrom89]. Device parameters depend on device size, orientation, and nearby polysilicon density. Therefore, for good matching it is best to build identical, relatively large transistors oriented in the same direction. For the same total gate area WL, long-channel transistors match better, so sense amplifiers can be built with longer than minimum devices [Lovett98]. When matching is especially critical, as in clock buffers, you can surround the transistor with a consistent pattern of polysilicon so all transistors see comparable nearby polysilicon densities. Threshold voltage variations are primarily caused by statistical fluctuations in the number of dopant atoms in the channel [Mizuno94]. As transistors shrink, they contain fewer dopant atoms and the statistical fluctuations become more severe.

Matching problems can be characterized as systematic, random, drift, or jitter. Systematic mismatches include factors that can be modeled and simulated at design time, such as wires of different lengths. Random mismatches include most process variations (length, threshold, and interconnect) that are either truly random or too costly to model. These mismatches do not change with time, so they can be nulled out through feedback circuits that detect the variation and compensate. Drift mismatches, notably temperature variation, change slowly with time as compared to the operating frequency of the system. Drift can again be nulled by compensation circuits, but such circuits must sample repeatedly faster than the drift occurs rather than just once at manufacturing or startup. Jitter, often from voltage variations, is the most difficult cause of mismatch. It occurs at frequen-

cies comparable to or faster than the system clock and therefore may not be eliminated through feedback.

As an example, [Harris01b] presents process variation data used to model the clock distribution network of the Itanium 2 processor in the Intel 180 nm process [Yang98]. The primary sources of random mismatch are channel lengths and threshold voltages. The channel length has both a component slowly varying across the die and a random component with no apparent spatial correlation. The slowly varying component varies over a 12.5 nm half-range for transistors separated by 4 mm or more; nearby transistors see less variation and adjacent transistors see no variation. The random component is modeled with a standard deviation of 3.3 nm. The threshold voltage variation has a Gaussian distribution with an inverse area dependence. It is treated as 16.8 mV for small nMOS transistors, 14.6 mV for small pMOS transistors, 7.9 mV for large nMOS transistors, and 6.5 mV for large pMOS transistors. Large transistors are defined as those with width exceeding 12.5 μm.

4.7.6 Delay Tracking



Another common design problem is how to build *matched delays*; for example, clock-delayed domino (see Section 7.5.4.2) needs to provide clocks to gates after their inputs have settled. The clocks must be matched to the gate delay; if they arrive late, the system functions slower, but if they arrive early, the system doesn't work at all. Therefore, it is of great interest to the designer how well two delays can be matched.

The best way to build matched delays is to actually provide replicas of the gates that are being matched. For example, in a static RAM (see Section 11.2.3), replica bitlines are used to determine when the sense amplifier should fire. Any relative variation in wire, diffusion, and gate capacitances happens to both circuits.

In many situations, it is not practical to use replica gates; instead, a chain of inverters can be used. Unfortunately, even if there is no intra-die process variation, the inverter delay may not exactly track the delay it matches across design corners. For example, if the inverter chain were matching a wire delay in the typical corner, it would be faster than the wire in the FFSFF corner and slower than the wire in the SSFSS corner. This variation requires that the designer provide margin in the typical case so that even in the worst case, the matched delay does not arrive too early. How much margin is necessary?

Figure 4.60 shows how gate delays, measured as a multiple of an FO4 inverter delay, vary with process, design corners, temperature, and voltage. The circuits studied include complementary CMOS NAND and NOR gates, domino AND and OR gates, and a 64-bit domino adder with significant wire RC delay. Figure 4.60(a) shows the gate delay of various circuits in different processes. The adder shows the greatest variation because of its wire-limited paths, but all the circuits track to within 20% across processes. This indicates that if a circuit delay is measured in FO4 inverter delays for one process, it will have a comparable delay in a different process. Figure 4.60(b and c) shows gate delay scaling with power supply voltage and temperature. Figure 4.60(d) shows what combination of design corner, voltage, and temperature gives the largest variation in delay normalized to an FO4 inverter in the same combination in the 0.6 μm process. Observe that the variation is

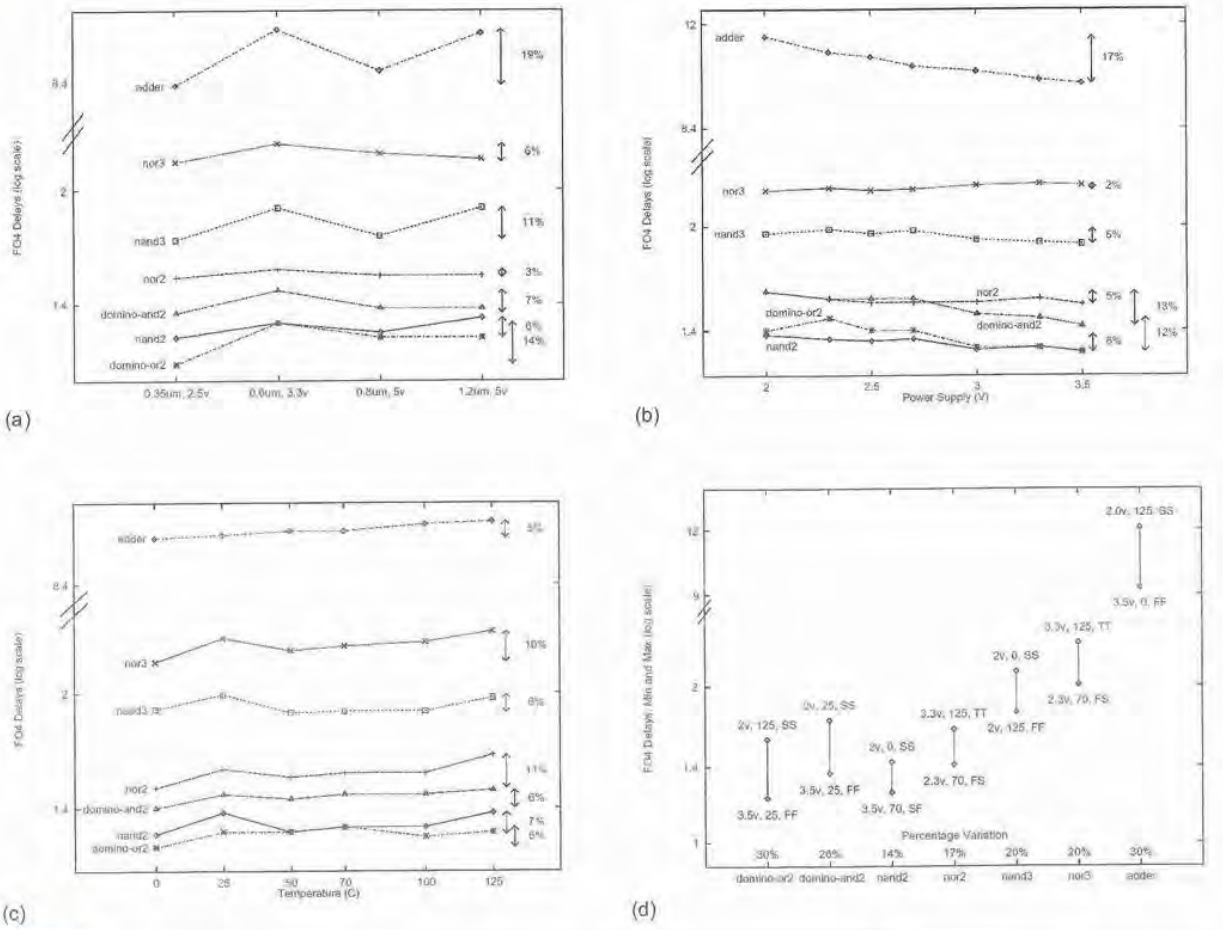


FIG 4.60 Delay tracking

smallest for simple static CMOS gates that most closely resemble inverters and can reach 30% for some gates. These figures demonstrate that an inverter chain should have a nominal delay about 30% greater than the path it matches so that the inverter output always arrives later than the matched path across all combinations of voltage, temperature, and design corners. This is a hefty margin and discourages the casual use of matched delays.

4.8 Reliability

Designing reliable CMOS chips involves understanding and addressing the potential failure modes [Greenhill02, Bernstein99]. This section addresses reliability problems (*hard errors*) that cause integrated circuits to fail permanently, including:

- Electromigration
- Self-heating
- Hot carriers
- Latchup
- Overvoltage failure

This section also considers transient failures (*soft errors*) that cause the system to crash or lose data. Circuit pitfalls and common design errors are discussed in Section 6.3.

4.8.1 Reliability Terminology

A number of acronyms are commonly used in describing reliability [Tobias95]. MTBF is the *mean time between failures*: (# devices • hours of operation) / # failures. FIT is the *failures in time*, the number of failures that would occur every thousand hours per million devices, or equivalently, 10^9 • (failure rate/hour). 1000 FIT is one failure in 10^6 hours = 114 years. This is good for a single chip. However, if a system contains 100 chips each rated at 1000 FIT and a customer purchases 10 systems, the failure rate is 100 • 1000 • 10 = 10^6 FIT, or one failure every 1000 hours (42 days). Reliability targets of less than 100 FIT are desirable.

Most systems exhibit the *bathtub curve* shown in Figure 4.61. Soon after birth, systems with weak or marginal components tend to fail. This period is called *infant mortality*. Reliable systems then enter their *useful operating life*, in which the failure rate is low. Finally, the failure rate increases at the end of life as the system wears out. It is important to age systems past infant mortality before shipping the products. Aging is accelerated by stressing the part through *burn-in* at higher than normal voltage and temperature. Nodes must toggle during burn-in to stress all parts of the system. Therefore, the circuits must be functional even under burn-in conditions.

Systems are also subjected to accelerated life testing during burn-in conditions to simulate the aging process and evaluate the time to wearout. The results are extrapolated to normal operating conditions to judge the actual useful operating life. This process is time-consuming and comes right at the end of the project. Part of any high-volume chip design will necessarily include designing a reliability assessment program that consists of burn-in boards deliberately stressing a number of chips over an extended period. Designers have tried to develop reliability simulators to predict lifetime [Hu92, Hsu92], but physical testing remains important. For high-volume parts, the source of failures is tracked and common points of failure can be redesigned and rolled into manufacturing.

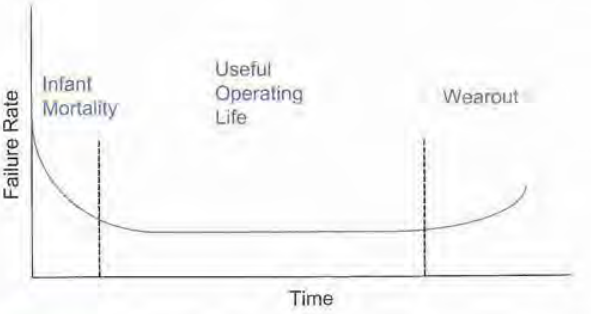


FIG 4.61 Reliability bathtub curve

4.8.2 Electromigration

Electromigration causes wearout of metal interconnect through the formation of voids [Hu95]. High current densities lead to an "electron wind" that causes metal atoms to migrate over time. Remarkable videos taken under a scanning electron microscope show void formation and migration and wire failure [Meier99]. The problem is especially severe for aluminum wires; it is commonly alleviated with an Al-Cu or Al-Si alloy and is much less important for pure copper wires because of the different grain transport properties. The electromigration properties also depend on the grain structure of the metal film.

Electromigration depends on the current density $J = I/wt$. It is more likely to occur for wires carrying a DC current where the electron wind blows in a constant direction than for those with bidirectional currents [Liew90]. Electromigration current limits are usually expressed as a maximum J_{dc}. The mean time to failure (MTTF) also is highly sensitive to operating temperature as given by Black's Equation [Black69]:

$$MTTF \propto \frac{e^{\frac{E_a}{kT}}}{J_{dc}^{\ n}} \tag{4.72}$$

where E_a is the activation energy that can be experimentally determined by stress testing at high temperatures and n is typically 2. The electromigration DC current limits vary with materials, processing, and desired MTTF and should be obtained from the fabrication vendor. In the absence of better information, a maximum J_{dc} of 1–2 mA/μm² is a conservative limit for aluminum wires at 110°C [Rzepka98], although 10 mA/μm² or better may be achievable for copper wires [Young00]. Current density may be even more limited in contact cuts.

4.8.3 Self-heating

While bidirectional wires are less prone to electromigration, their current density is still limited by *self-heating*. High currents dissipate power in the wire, which raises its ambient temperature. Hot wires exhibit greater resistance and delay. Electromigration is also highly sensitive to temperature, so self-heating may cause temperature-induced electromigration problems in the bidirectional wires. Brief pulses of high peak currents may even melt the interconnect. Self-heating is dependent on the RMS current density. This can be measured with a circuit simulator or calculated as

$$I_{rms} = \sqrt{\frac{\int_0^T I(t)^2 dt}{T}} \tag{4.73}$$

A reasonable rule to control reliability problems with self-heating is to keep $J_{rms} < 15$ mA/μm² for bidirectional aluminum wires [Rzepka98] on a silicon substrate. Self-heating is especially significant for SOI processes because of the poor thermal conductivity of SiO_2.

In summary, electromigration from high DC current densities is primarily a problem in power and ground lines. Self-heating limits the RMS current density in bidirectional signal lines. However, do not overlook the significant unidirectional currents that flow through the wires contacting nMOS and pMOS transistors. For example, Figure 4.62 shows which lines in an inverter are limited by DC and RMS currents. Both problems can be addressed by widening the lines or reducing the transistor sizes (and hence the current).

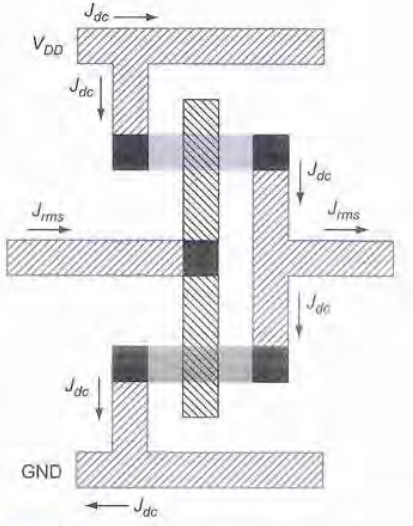


FIG 4.62 Current density limits in an inverter

4.8.4 Hot Carriers

As transistors switch, some high-energy ("hot") carriers may be injected into the gate oxide and become trapped there. The damaged oxide changes the I-V characteristics of the device, reducing current in nMOS transistors and increasing current in pMOS transistors. Damage is maximized when the substrate current I_{sub} is large, which typically occurs when nMOS transistors are in saturation while the input rises. Therefore, the problem is worst for inverters and NOR gates with fast rising inputs and heavily loaded outputs [Sakurai86], and for high power supply voltages.

Hot carriers cause circuit wearout as nMOS transistors become too slow. They can also cause failures of sense amplifiers and other matched circuits if matched components degrade differently [Huh98]. Hot electron degradation can be analyzed with simulators [Hu92, Hsu91, Quader94]. The wear is limited by setting maximum values on input rise-time and stage electrical effort [Leblebici96]. These maximum values depend on the process and operating voltage.

A related aging mechanism is *negative bias temperature instability* (NBTI), which leads to a decrease in pMOS transistor current as transistors wear at high temperature [Doyle91]. NBTI results from trapped holes in the oxide coupled with the creation of interface states. Like hot carriers, it leads to circuit failures from increased delay and poorer matching [Reddy02]. NBTI shifts depend on the electric field seen by the device and can be locked in to the device by high-voltage stress during burn-in; this is good because it allows testing with full NBTI degradation.

4.8.5 Latchup

Early adoption of CMOS processes was slowed by a curious tendency of CMOS chips to develop low-resistance paths between V_{DD} and GND, causing catastrophic meltdown. The phenomenon, called *latchup*, occurs when parasitic bipolar transistors formed by the substrate, well, and diffusion turn ON. With process advances and proper layout procedures, latchup problems can be easily avoided.

The cause of the latchup effect [Estreich82, Troutman86] can be understood by examining the process cross-section of a CMOS inverter, shown in Figure 4.63(a), over which is overlaid an equivalent circuit. In addition to the expected nMOS and pMOS transistors, the schematic depicts a circuit composed of an npn-transistor, a pnp-transistor, and two resistors connected between the power and ground rails (Figure 4.63(b)). The npn-transistor is formed between the grounded n-diffusion source of the nMOS transistor, the p-type substrate, and the n-well. The resistors are due to the resistance through the substrate or well to the nearest substrate and well taps. The cross-coupled transistors form a bistable silicon-controlled rectifier (SCR). Ordinarily, both parasitic bipolar transistors are OFF. Latchup can be triggered when transient currents flow through the substrate during normal chip power-up or when external voltages outside the normal operating range are applied. If substantial current flows in the substrate, V_{sub} will rise, turning ON the npn-transistor. This pulls current through the well resistor, bringing down V_{well} and turning ON the pnp-transistor. The pnp-transistor current in turn raises V_{sub}, initiating a positive feedback loop with a large current flowing between V_{DD} and GND that persists until the power supply is turned off or the power wires melt.

Fortunately, latchup prevention is easily accomplished by minimizing R_{sub} and R_{well}. Some processes use a thin epitaxial layer of lightly doped silicon on top of a heavily doped substrate that offers a low substrate resistance. Most importantly, the designer should place substrate and well taps close to each transistor, as described in Section 1.5.1. A conservative guideline is to place a tap adjacent to every source connected to V_{DD} or GND. If this is not practical, you can obtain more detailed information from the process vendor (they will normally specify a maximum distance for diffusion to substrate/well tap) or try the following guidelines:

- Every well should have at least one tap.
- All substrate and well taps should connect directly to the appropriate supply in metal.

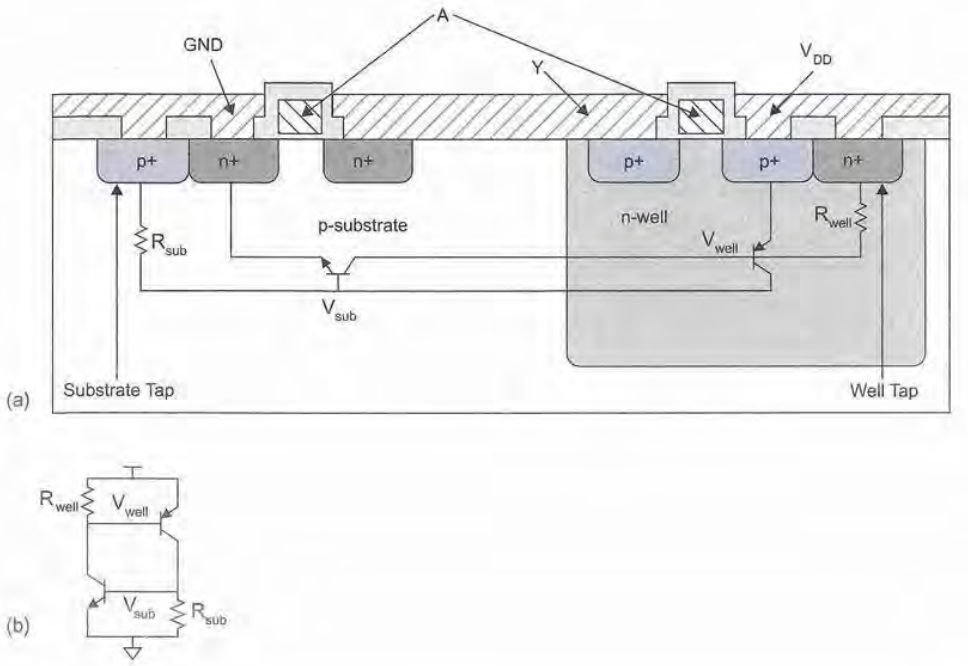


FIG 4.63 Origin and model of CMOS latchup

- A tap should be placed for every 5-10 transistors or every 25-100 μm (this distance is process-dependent).
- nMOS transistors should be clustered together near GND and pMOS transistors should be clustered together near V_{DD}, avoiding convoluted structures that intertwine nMOS and pMOS transistors in checkerboard patterns.

I/O pads are especially susceptible to latchup because external voltages can ring below GND or above V_{DD}, forward biasing the junction between the drain and substrate or well and injecting current into the substrate. In such cases, guard rings should be used to collect the current, as shown in Figure 4.64. Guard rings are simply substrate or well taps tied to the proper supply that completely surround the transistor of concern. For example, the n+ diffusion in Figure 4.64(b) can inject electrons into the substrate if it falls a diode drop below 0 volts. The p+ guard ring tied to ground provides a low-resistance path to collect these electrons before they interfere with the operation of other circuits outside the guard ring. *All* diffusion structures in any circuit connected to the external world must be guard ringed, i.e., n+ diffusion by p+ connected to GND or p+ diffusion by n+ connected to V_{DD}.

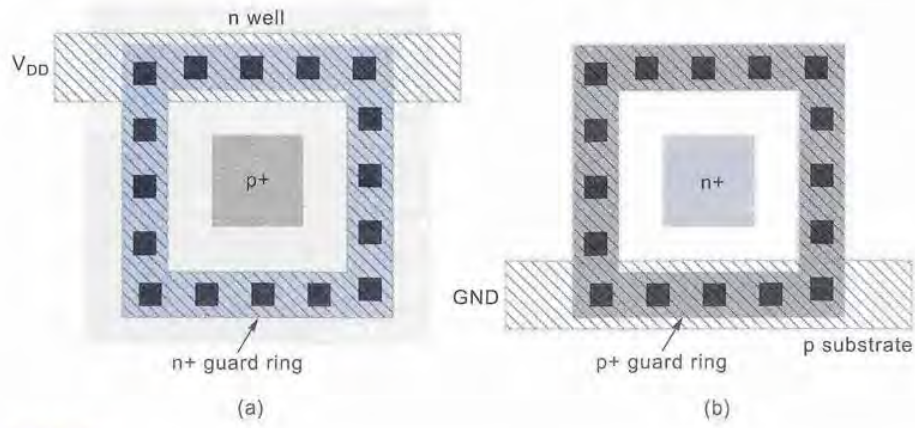


FIG 4.64 Guard rings

For the ultra-paranoid, double guard rings may be employed, i.e., n+ ringed by p+ to GND, then n+ to V_{DD} or p+ ringed by n+ to V_{DD}, then p+ to GND.

SOI processes avoid latchup entirely because they have no parasitic bipolar structures. Also, processes with $V_{DD} < 0.7$ are immune to latchup because the parasitic transistors will never have a large enough base-to-emitter voltage to turn on. In general, low-voltage processes are much less susceptible to latchup problems.

4.8.6 Overvoltage Failure

Tiny transistors can be easily damaged by relatively low voltages. *Overvoltage* reliability problems can arise from electrostatic discharge, oxide breakdown, punchthrough, and time-dependent dielectric breakdown of the gate oxide. *Electrostatic discharge* (ESD) from static electricity entering the I/O pads can cause very large voltage and current transients and is discussed further in Section 12.4.

Undesired voltages applied to the gate can cause *breakdown* and *arcing* across the thin oxide, destroying the device. Higher-than-normal voltages applied between source and drain lead to *punchthrough* when the source/drain depletion regions touch. Both problems lead to a maximum safe voltage that can be applied to transistors. For modern processes, this voltage is often much less than the I/O standard voltage, requiring a second type of transistor with thicker oxides and longer channels to endure the higher I/O voltages.

Gate oxides wear out with time as tunneling currents cause irreversible damage to the oxide; this problem is called *time-dependent dielectric breakdown* (TDDB). The failure rate is exponentially dependent on the temperature and oxide thickness; for a 10-year life at 125° C, the field across the gate $E_{ox} = V_{DD}/t_{ox}$ should be kept below about 7 MV/cm = 0.7 V/nm [Moazzami90]. The problem is greatest when voltage overshoots occur; this can

be caused by noisy power supplies or reflections at I/O pads. Reliability is improved by lowering the power supply voltage, minimizing power supply noise, and using thicker oxides on the I/O pads.

4.8.7 Soft Errors

In the 1970s, as dynamic RAMs (DRAMs) replaced core memories, DRAM vendors were puzzled to find DRAM bits occasionally flipping value spontaneously. At first, the errors were attributed to "system noise," "voltage marginality," "sense amplifiers," or "pattern sensitivity," but the errors were found to be random. When the corrupted bit was rewritten with a new value, it was no more likely than any other bit to experience another error. In a classic paper [May79], Intel identified the source of these soft errors as alpha particle collisions that generate electron-hole pairs in the silicon as the particles lose energy. The excess carriers can be collected into the diffusion terminals of transistors. If the charge collected is comparable to the charge on the node, the voltage can be disturbed.

Soft errors are random nonrecurring single bit errors in memory devices, including SRAM, DRAM, registers, and latches. Alpha particles from decaying uranium and thorium impurities in integrated circuit interconnect and packaging is a major source of soft errors at sea level, often causing a *soft error rate* (SER) of 100–2000 FIT/Mb [Hazucha00]. The neutron flux from cosmic rays is low at sea level, but two orders of magnitude larger at aircraft flight altitudes [Ziegler96]. These cosmic rays cause up to 10^6 FIT/Mb at flight altitudes.

Soft errors are minimized by maintaining at least some critical charge Q_{crit} ($Q = CV$) on state nodes. This is difficult as memory cells become smaller and hence have less capacitance, and as power supplies get lower. DRAMs use exotic structures such as trench capacitors to hold sufficient charge. Fortunately, small cells require less Q_{crit} because of their small area to collect carriers.

The error rate is also affected by the proximity of alpha sources. Flip-chip technology with solder bumps bonded directly to the die poses special risks because impurities in the lead bumps emit a high alpha particle flux. Some companies avoid placing solder bumps directly over RAMs or other sensitive circuits. Aged lead (e.g., from Roman pipes) has outlived the contaminant half-lives and is much safer. Similarly, highly purified aluminum interconnect reduces the alpha flux from chip wires [Juhnke95].

Finally, error detecting and correcting codes can be used to tolerate soft errors in memories without data corruption. These codes will be discussed further in Section 10.7.2.

4.9 Scaling

The only constant in VLSI design is constant change. Figure 4.65 shows the year of introduction of Intel microprocessors in each feature size on a logarithmic scale, indicating that feature size reduces by 30% from one generation to the next every 2 to 3 years. As transis-

tors become smaller, they switch faster, dissipate less power, and are cheaper to manufacture! Despite the ever-increasing challenges, process advances have actually accelerated in the past decade. Such scaling is unprecedented in the history of technology. However, scaling also exacerbates noise and reliability issues and introduces new problems. Designers need to be able to predict the effect of this feature size scaling on chip performance to plan future products, ensure existing products will scale gracefully to future processes for cost reduction, and anticipate looming design challenges. This section examines how transistors and interconnect scale, and the implications of scaling for design. The Semiconductor Industry Association prepares and maintains an International Technology Roadmap for Semiconductors predicting future scaling. Section 4.11 gives a case study of how scaling has influenced Intel microprocessors over three decades.

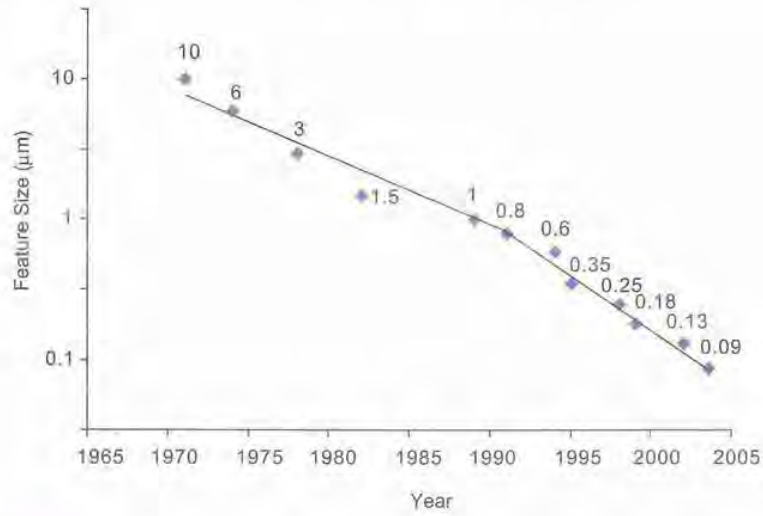


FIG 4.65 Year of introduction of processes

4.9.1 Transistor Scaling

First-order *constant field* MOS scaling theory is based on a model formulated by Dennard [Dennard74]. The characteristics of an MOS device can be maintained and the basic operational characteristics can be preserved if the critical parameters of a device are scaled by a dimensionless factor S. These parameters include

- All dimensions (in the x, y, and z directions)
- Device voltages
- Doping concentration densities

Another approach is *lateral scaling*, in which only the gate length is scaled. This is commonly called a *gate shrink* because it can be done easily to an existing mask database for a design.

The effects of these types of scaling are illustrated in Table 4.15. The industry generally scales process generations with $S = \sqrt{2}$. This doubles the number of transistors per unit area with each generation and doubles transistor performance every two generations under constant field scaling. A 5% gate shrink (S = 1.05) is commonly applied as a process becomes mature to boost the speed of components in that process.

Table 4.15 Influence of scaling on MOS device characteristics

Parameter	Sensitivity	Constant Field	Lateral
Scaling Parameters			
Length: L		$1/S$	$1/S$
Width: W		$1/S$	1
Gate oxide thickness: t_{ox}		$1/S$	1
Supply voltage: V_{DD}		$1/S$	1
Threshold voltage: V_{tn}, V_{tp}		$1/S$	1
Substrate doping: N_A		S	1
Device Characteristics			
β	$\frac{W}{L}\frac{1}{t_{ox}}$	S	S
Current: I_{ds}	$\beta(V_{DD} - V_t)^2$	$1/S$	S
Resistance: R	$\frac{V_{DD}}{I_{ds}}$	1	$1/S$
Gate capacitance: C	$\frac{WL}{t_{ox}}$	$1/S$	$1/S$
Gate delay: τ	RC	$1/S$	$1/S^2$
Clock frequency: f	$1/\tau$	S	S^2
Dynamic power dissipation (per gate): P	CV^2f	$1/S^2$	S
Chip area: A		$1/S^2$	1
Power density	P/A	1	S
Current density	I_{ds}/A	S	S

For constant field scaling, all device dimensions, including channel length L, width W, and oxide thickness t_{ox}, are reduced by a factor of $1/S$. The supply voltage V_{DD} and the threshold voltages are also reduced by $1/S$. The substrate doping N_A is increased by S. Because both distance and voltage are scaled equally, the electric field remains constant. This has the desirable effect that many nonlinear factors essentially remain unaffected. Because both L and t_{ox} scale at the same rate, the gate capacitance per micron of width $C_{permicron}$ has also remained approximately constant at 1.5–2 fF/μm, as described in Section 2.3.1.

A gate shrink scales only the channel length, leaving other dimensions, voltages, and doping levels unchanged. This offers a quadratic improvement in gate delay according to the first order model. In practice, the gate delay improvement is closer to linear because velocity saturation keeps the current and effective resistance approximately constant.

Historically, feature sizes were shrunk from 6 μm to 1 μm while maintaining a 5 V supply voltage. This was called *constant voltage* scaling and offered quadratic delay improvement as well as cost reduction. It also maintained continuity in I/O voltage standards. Constant voltage scaling increased the electric fields in devices. By the 1 μm generation, velocity saturation was severe enough that decreasing feature size no longer improved device current. Device breakdown from the high field was another risk. Therefore, constant field scaling has been the rule for modern devices.

Example

Most processes have gate capacitance of roughly 2 fF/μm. If the FO4 inverter delay of a process with features size f (in nm) is 1/2 ps • f, estimate the ON resistance of a unit (i.e., 4 λ wide) nMOS transistor.

Solution: An FO4 inverter has a delay of 5 τ = 15 RC. Therefore,

$$RC = \frac{\frac{1}{2}f}{15} = \frac{f}{30}\frac{\text{ps}}{\text{nm}} \tag{4.74}$$

A unit transistor has width $W = 2f$ and thus capacitance of $C = 4f$ fF/μm. Solving for R,

$$R = \left(\frac{f}{30}\frac{\text{ps}}{\text{nm}}\right)\left(\frac{1}{4f}\frac{\mu\text{m}}{\text{fF}}\right) = 8.33\ \text{k}\Omega \tag{4.75}$$

Note that this is independent of feature size. The resistance of a unit transistor is roughly independent of feature size, while the gate capacitance decreases with feature size. Alternatively, the capacitance per micron is roughly independent of feature size while the resistance • micron decreases with feature size.

The FO4 inverter delay will scale as $1/S$ assuming ideal constant-field scaling. As we saw in Section 4.7.4, this delay is commonly 1/2 ps/nm of the effective channel length for typical processing and worst-case environment. Aggressive processes achieve delays in the short end of the range by building transistors with effective channel lengths somewhat shorter than the feature size would imply.

4.9.2 Interconnect Scaling

Two common approaches to interconnect scaling are to either scale all dimensions or keep the wire height constant. Table 4.16 shows the resistance, capacitance, and delay per unit length for each of these techniques. Wire length decreases for some types of wires, but may increase for others. Local and scaled wires are those that decrease in length during scaling. For example, a wire across a 64-bit ALU is *local* because it becomes shorter as the ALU is migrated to a finer process. A wire across a particular microprocessor is *scaled* because when the microprocessor is shrunk to the new process, the wire will also shrink. However, wires crossing next generation microprocessors are *global* because the next generation microprocessor die is likely to be larger (by a factor of D_c, on the order of 1.1) than the previous generation, so the cross-chip wires get longer rather than shorter. The table also shows scaling of wire delay for interconnect with and without repeaters.

Unrepeated interconnect delay is remaining about constant for local interconnect and increasing for global interconnect. This presents a problem because transistors are getting faster, so the ratio of interconnect to gate delay increases with scaling. An increasing fraction of circuits are limited by wire delay rather than gate delay, and designers must devote greater attention to wire engineering and repeaters.

In older processes where wire width and spacing were much greater than wire thickness, it was advantageous to scale wires by reducing the width and spacing, but not the thickness. This avoids the quadratic increase in wire resistance per unit length and was acceptable because fringing capacitance was a small fraction of the whole. In modern processes with aspect ratios of 1.5–2.2, fringing capacitance accounts for the majority of the total capacitance. Scaling spacing but not height increases the fringing capacitance enough that the extra thickness scarcely improves delay. Moreover, the coupling capacitance to nearby neighbors causes severe crosstalk. Therefore, it is now common to reduce thickness of lower-level metal interconnect with each generation. Of course, process engineers can choose from a continuum of possibilities between linearly scaling wire thickness and keeping wire thickness constant.

Observe that when wire thickness is scaled, the capacitance per unit length remains constant. Hence, a reasonable initial estimate of the capacitance of a minimum-pitch wire is about 0.2 fF/μm, independent of the process. In other words, wire capacitance is roughly 1/10–1/6 of gate capacitance per unit length.

Table 4.16 Influence of scaling on interconnect characteristics

Parameter	Sensitivity	Reduced Thickness	Constant Thickness
Scaling Parameters			
Width: w		$1/S$	
Spacing: s		$1/S$	
Thickness: t		$1/S$	1
Interlayer oxide height: h		$1/S$	
Characteristics Per Unit Length			
Wire resistance per unit length: R_w	$\frac{1}{wt}$	S^2	S
Fringing capacitance per unit length: C_{wf}	$\frac{t}{s}$	1	S
Parallel plate capacitance per unit length: C_{wp}	$\frac{w}{h}$	1	1
Total wire capacitance per unit length: C_w	$C_{wf} + C_{wp}$	1	between 1, S
Unrepeated RC constant per unit length: t_{wu}	$R_w C_w$	S^2	between S, S^2
Repeated wire RC delay per unit length: t_{wr} (assuming constant field scaling of gates in Table 4.15)	$\sqrt{RCR_w C_w}$	$\sqrt{S}$	between 1, $\sqrt{S}$
Crosstalk noise	$\frac{t}{s}$	1	S
Local/Scaled Interconnect Characteristics			
Length: l		$1/S$	
Unrepeated wire RC delay	$l^2 t_{wu}$	1	between $1/S$, 1
Repeated wire delay	$l t_{wr}$	$\sqrt{1/S}$	between $1/S$, $\sqrt{1/S}$
Global Interconnect Characteristics			
Length: l		D_c	
Unrepeated wire RC delay	$l^2 t_{wu}$	$S^2 D_c^2$	between SD_c^2, $S^2 D_c^2$
Repeated wire delay	$l t_{wr}$	$D_c\sqrt{S}$	between D_c, $D_c\sqrt{S}$

4.9.3 International Technology Roadmap for Semiconductors

The incredible pace of scaling requires cooperation among many companies and researchers both to develop compatible process steps and to anticipate and address future challenges before they hold up production. The Semiconductor Industry Association develops and updates the International Technology Roadmap for Semiconductors [SIA02] to forge a consensus so that development efforts are not wasted on incompatible technologies and to predict future needs and direct research efforts. Such an effort to predict the future is inevitably prone to error, and the industry has scaled feature sizes and clock frequencies more rapidly than the roadmap predicted in the late 1990s. Nevertheless, the roadmap offers a more coherent vision than one could obtain by simply interpolating straight lines through historical scaling data.

The ITRS forecasts a major new technology generation, also called *technology node*, approximately every three years. The scaling between generations is traditionally

$$S = \sqrt{2}$$

so the number of transistors per unit area doubles every generation. Table 4.17 summarizes some of the predictions, particularly for high-performance microprocessors. However, serious challenges lie ahead, and major breakthroughs will be necessary in many areas to maintain the scaling on the roadmap. The FO4 delays/cycle figure is extracted assuming the FO4 delay in picoseconds is one-third of the feature size in nanometers. These cycle times appear problematic because it is very difficult for a digital signal to swing rail to rail in less than 6 FO4 delays. Refer to the ITRS for annual updates.

Table 4.17 Predictions from the 2002 ITRS

Year	2001	2004	2007	2010	2013	2016
Feature size (nm)	130	90	65	45	32	22
V_{DD} (V)	1.1–1.2	1–1.2	0.7–1.1	0.6–1.0	0.5–0.9	0.4–0.9
Millions of transistors/die	193	385	773	1564	3092	6184
Wiring levels	8–10	9–13	10–14	10–14	11–15	11–15
Intermediate wire pitch (nm)	450	275	195	135	95	65
Interconnect dielectric constant	3–3.6	2.6–3.1	2.3–2.7	2.1	1.9	1.8
I/O signals	1024	1024	1024	1280	1408	1472
Clock rate (MHz)	1684	3990	6739	11511	19348	28751
FO4 delays/cycle	13.7	8.4	6.8	5.8	4.8	4.7
Maximum power (W)	130	160	190	218	251	288
DRAM capacity (Gbits)	0.5	1	4	8	32	64

4.9.4 Impacts on Design

One of the limitations of first-order scaling is that it gives the wrong impression of being able to scale proportionally to zero dimensions and zero voltage. In reality, a number of factors change significantly with scaling. This section attempts to peer into the crystal ball and predict some of the impacts on design for the future. These predictions are notoriously risky because chip designers have had an astonishing history of inventing ingenious solutions to seemingly insurmountable barriers.

4.9.4.1 Improved Performance and Cost The most positive impact of scaling is that performance and cost are steadily improving. System architects need to understand the scaling of CMOS technologies and predict the capabilities of the process several years into the future, when a chip will be completed. Because transistors are becoming cheaper each year, architects particularly need creative ideas of how to exploit growing numbers of transistors to deliver more or better functions. When transistors were first invented, the best predictions of the day suggested that they might eventually approach a fifty-cent manufacturing cost. Figure 4.66 plots the number of transistors and average price per transistor shipped by the semiconductor industry over the past three decades [Moore03]. In 2003, you could buy more than 100,000 transistors for a penny.

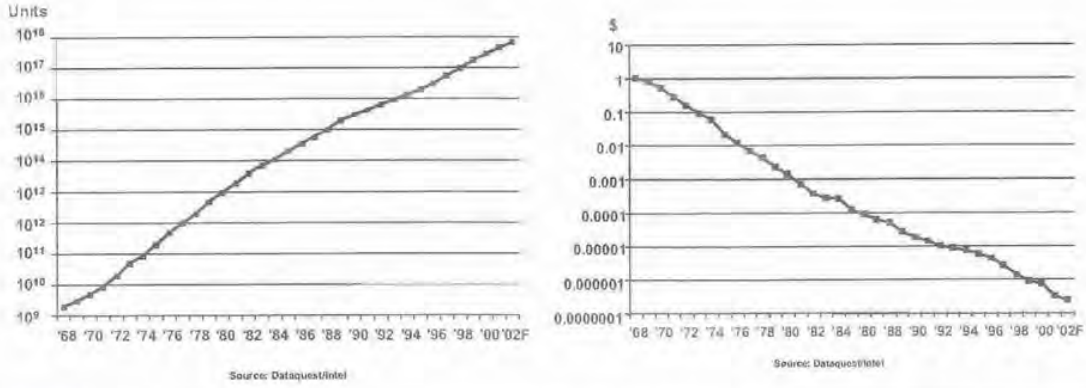


FIG 4.66 Transistor shipments and average price. © IEEE 2003.

4.9.4.2 Interconnect Scaled transistors are steadily improving in delay, but scaled wires are holding constant or getting worse. Figure 4.67, taken from the 1997 Semiconductor Industry Association Roadmap [SIA97], showed the sum of gate and wire bottoming out at the 250 or 180 nm generation and getting worse thereafter. The wire problem motivated a number of papers predicting the demise of conventional wires. However, the plot

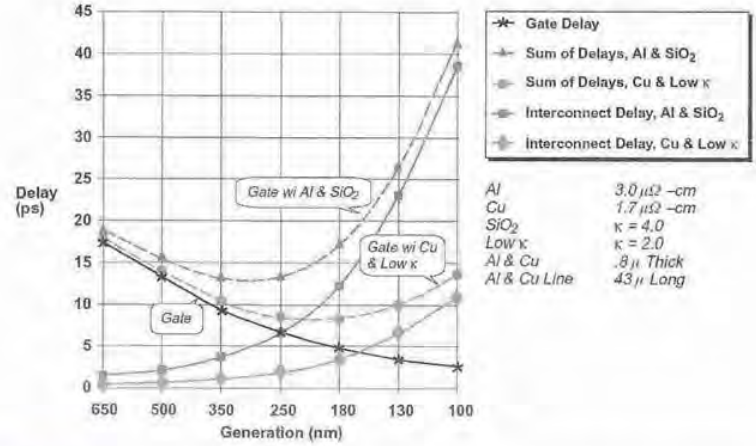


FIG 4.67 Gate and wire delay scaling. Reprinted from [SIA97] with permission of the Semiconductor Industry Association.

is misleading in two ways. First, the "gate" delay is shown for a single unloaded transistor (delay = *RC*) rather than a realistically loaded gate (e.g., an FO4 inverter delay = 15*RC*). Second, the wire delays shown are for fixed lengths, but as technology scales, most local wires connecting gates within a unit also become shorter [Ho01].

In practice, for short wires, such as those inside a logic gate, the wire RC delay is negligible and will remain so for the foreseeable future. However, the long wires present a considerable challenge. It is no longer possible to send a signal from one side of a large, high-performance chip to another in a single cycle. Also, the "reachable radius" that a signal can travel in a cycle is steadily getting smaller, as shown in Figure 4.68. This requires that microarchitects understand the floorplan and budget multiple pipeline stages for data to travel long distances across the die.

Repeaters help somewhat, but even so, interconnect does not keep up. Moreover, the "repeater farms" must be allocated space in the floorplan. As scaled gates become faster, the delay of a repeater goes down and hence, you should expect it will be better to use more repeaters. This means a greater number of repeater farms are required.

One technique to alleviate the interconnect problem is to use more layers of interconnect. Table 4.18 shows the number of layers of interconnect increasing with each generation in TSMC processes. The lower layers of interconnect are classically scaled to provide high-density short connections. The higher layers are scaled less aggressively, or possibly even reverse-scaled to be thicker and wider to provide low-resistance, high-speed interconnect, good clock distribution networks, and a stiff power grid. Copper and low-k dielectrics were also introduced to reduce resistance and capacitance.

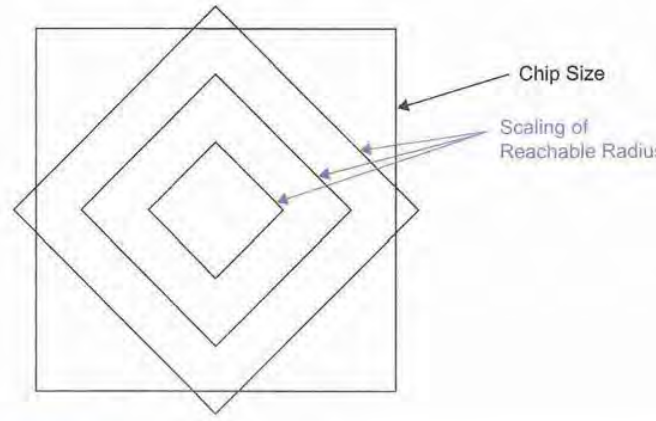


FIG 4.68 Reachable radius scaling

Table 4.18 Scaling of metal layers in TSMC processes

Process (nm)	Metal Layers
500	3 (Al)
350	4 (Al)
250	5 (Al)
180	6 (Al, low-k)
150	7 (Cu, low-k)
130	8 (Cu, low-k)
90	9 (Cu, low-k)

Blocks of 50–100 Kgates (1 *Kgate* = 1000 3-input NAND gates or 6000 transistors) will continue to have reasonably short internal wires and acceptably low wire RC delay [Sylvester98]. Therefore, large systems can be partitioned into blocks of roughly this size with repeaters inserted as necessary for communication between blocks.

4.9.4.3 Power In classical constant field scaling, power density remains constant and overall chip power increases only slowly with die size. In practice, power density has skyrocketed because clock frequencies have increased much faster than classical scaling would predict and V_{DD} is somewhat higher than constant field scaling would demand.

Intel Vice President Patrick Gelsinger gave a keynote speech at the International Solid State Circuits Conference on February 5, 2001 [Gelsinger01]. He showed that microprocessor power consumption had been increasing exponentially (see Figure 4.69)

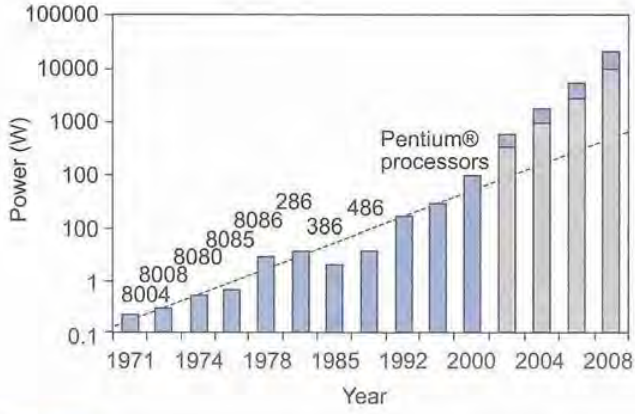


FIG 4.69 Intel processor power consumption. © IEEE 2001.

and was predicted to grow even faster in the coming decade. He predicted "business as usual will not work in the future," and that if scaling continued at this pace, by 2005, high-speed processors would have the power density of a nuclear reactor, by 2010, a rocket nozzle, and by 2015, the surface of the sun! The next day, Intel's stock dropped 8% and Intel has since downplayed the forecast because designers will obviously devote more attention to controlling power.

Dynamic power consumption will not continue to increase at such rates because it will become uneconomical to cool the chips. One reason is that at the time of this writing, high-performance processors are operating at about 12–16 FO4 inverter delays/cycle. It will be very difficult to generate a clock with a period of less than 6 FO4 inverter delays because the clock will look like a sinusoid rather than a square wave [Ho01] and because sequencing overhead becomes excessive; hence, frequency cannot scale faster than raw gate delays too much longer. Another reason is that cache area is likely to become a larger fraction of the die area and caches have low activity factors and lower power dissipation per unit area. Nevertheless, designers will need to budget and plan for power consumption as a factor nearly as important as performance and perhaps more important than area.

Static power consumption will be a growing concern, especially for battery-operated devices. The static power consumption caused by subthreshold leakage was historically negligible but becomes important for threshold voltages below about 0.3–0.4 V, and may become comparable to dynamic power for systems operating below 1 V. Figure 4.70 shows how the static power has historically increased much more rapidly than dynamic power [Moore03]. This is especially problematic because simply turning off clocks in a sleep mode is not sufficient to stop the static power consumption. To slow the increase of leakage, V_t has changed from about $V_{DD}/5$ in older processes to $V_{DD}/3$ in newer processes as

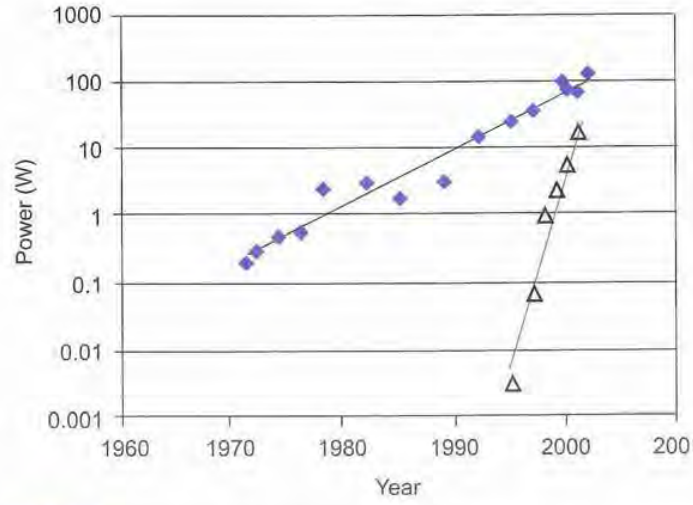


FIG 4.70 Dynamic and static power trends. © IEEE 2003.

V_{DD} has dropped; this has eliminated circuits with threshold drops from the set of viable circuit styles. Systems will use multiple flavors of transistors with different threshold voltages so that gates on the critical path can have fast low-V_t devices while memories and noncritical gates save power with higher-V_t devices. Gate tunneling current is also important for oxides of less than 15–20 Å. Multiple oxide thicknesses are also required. I/O standards of 1.8 or 3.3 V are destructive to the very thin oxides in modern processes, so thicker oxides are used for slower I/O transistors. Thicker oxides may also become an option for memories and noncritical circuits.

Even if power remains constant, lower supply voltage leads to higher current density. This in turn causes higher IR drops and di/dt noise in the supply network (see Sections 6.3.5 and 12.3). These factors lead to more pins and metal resources on a chip being required for the power distribution networks.

4.9.4.4 Productivity The number of transistors that fit on a chip is increasing faster than designer productivity (gates/week). This leads to design teams of increasing size, and difficulty recruiting enough experienced engineers when the economy is good. It has driven a search for design methodologies that maximize productivity, even at the expense of performance and area. Now most chips are designed using synthesis and place and route; the number of situations where custom circuit design is affordable is diminishing. In other words, creativity is shifting from the circuit to the systems level for many designs. On the other hand, performance is still king in the microprocessor world. Design teams in that

field are approaching the size of automotive and aerospace teams because the development cost is justified by the size of the market. This drives a need for engineering managers who are skilled in leading such large organizations.

The number of 50–100 Kgate blocks is growing, even in relatively low-end systems. This demands greater attention to floorplanning and placement of the blocks.

One of the best hopes to solve the productivity gap is design reuse. Intellectual property (IP) blocks can be purchased and used as black boxes within a system-on-chip (SoC) in much the same way chips are purchased for a board-level design.

4.9.4.5 Physical Limits How far will CMOS processes scale? It is clear that scaling cannot continue indefinitely; transistors as we know them today will not work if the oxide is less than an atomic layer thick, the channel less than an atomic layer long, or the charge in the channel less than that of one electron. Numerous papers have been written forecasting the end of silicon scaling. For example, in 1972, the limit was placed at the 0.25 μm generation because of tunneling and fluctuations in dopant distributions [Hoeneisen72, Mead80]; at this generation, chips were predicted to operate at 10–30 MHz! In 1999, IBM predicted that scaling would nearly grind to a halt beyond the 100 nm generation in 2004 [Davari99].

In the authors' experience, seemingly insurmountable barriers have seemed to loom about a decade away. Reasons given for these barriers have included:

- Dynamic power dissipation
- Subthreshold leakage at low V_{DD} and V_t
- Tunneling current through thin oxides
- Poor I-V characteristics due to short channel effects
- Optics for cost-effective manufacturing of small features
- Exponentially increasing costs of fabrication facilities and mask sets
- Electromigration
- Interconnect delay

At the time of writing, it appears that no fundamental barriers exist before the 35 nm generation in 2013 (roughly a decade ahead). Beyond this point, it is difficult to predict the future. Nevertheless, a large number of extremely talented people are continuously pushing the limits and hundreds of billions of dollars are at stake, so we are reluctant to bet against the future of scaling.

4.10 Pitfalls and Fallacies

Defining gate delay for an unloaded gate

When marketing a process, it is common to report gate delay based on an inverter in a ring oscillator (2τ) or even the RC time constant of a transistor charging its own gate capacitance (1/3 τ). Remember that the delay of a real gate on the critical path should be closer to 5–6τ. When in doubt, ask how "gate delay" is defined or ask for the FO4 inverter delay.

Trying to increase speed by increasing the size of transistors in a path

Most designers know that increasing the size of a transistor decreases its resistance and thus makes it faster at driving a constant load. Novice designers sometimes forget that increasing the size increases input capacitance and makes the previous stage slower, especially when that previous stage belongs to somebody else's timing budget. The authors have seen this lead to lack of convergence in full-chip timing analysis on a large microprocessor because individual engineers boost the size of their own gates until their path meets timing. Only after the weekly full-chip timing roll-up do they discover that their inputs now arrive later because of the greater load on the previous stage. The solution is to include in the specification of each block not only the arrival time but also the resistance of the driver in the previous block.

Trying to increase speed by using as few stages of logic as possible

Logic designers often count "gate delays" in a path. This is a convenient simplification when used properly. In the hands of an inexperienced engineer who believes each gate contributes a gate delay, it suggests that the delay of a path is minimized by using as few stages of logic as possible, which is clearly untrue.

Designing a large chip without considering the floorplan

In the mid-1990s, designers became accustomed to synthesizing a chip from HDL and "tossing the netlist over the wall" to the vendor who would place and route it and manufacture the chip. Many designers were shielded from considering the physical implementation. Now flight times across the chip are a large portion of the cycle time in slow systems and multiple cycles in faster systems. If the chip is synthesized without a floorplan, some paths with long wires will be discovered to be too slow after layout. This requires resynthesis with new timing constraints to shorten the wires. When the new layout is completed, the long wires simply show up in different paths. The solution to this convergence problem is to make a floorplan early and microarchitect around this floorplan, including budgets for wire flight time between blocks. Algorithms termed *timing directed placement* have alleviated this problem, resulting in place & route tools that converge in one or a few iterations.

Not stating process corner or environment when citing circuit performance

Most products must be guaranteed to work at high temperature, yet many papers are written with transistors operating at room temperature (or lower), giving optimistic performance results. For example, at the International Solid State Circuits Conference Intel described a Pentium II processor running at a surprisingly high clock rate [Choudhury97], but when asked, the speaker admitted that the measurements were taken while the processor was "colder than an ice cube."

Similarly, the FFFFF design corner is sometimes called the "published paper" corner because delays are reported under these simulation or manufacturing con-

ditions without bothering to state that fact or report the FO4 inverter delay in the same conditions. Circuits in this corner are about twice as fast as in a manufacturable part.

Providing too little margin in matched delays

We have seen that the delay of a chain of inverters can vary by about 30% as compared to the delay of other circuits across design corners, voltage, and temperature. On top of this, you should expect intra-die process variation and errors in modeling and extraction. If a race condition exists where the circuit will fail when the inverter delay is faster than the gate delay, the experienced designer who wishes to sleep well at night provides generous delay margin under nominal conditions. Remember that the consequences of too little margin can be a million dollars in mask costs for another revision of the chip and far more money in the opportunity cost of arriving late to market.

Failing to plan for process scaling

Many products will migrate through multiple process generations. For example, the Intel Pentium Pro was originally designed and manufactured on a 0.6 micron BiCMOS process. The Pentium II is a closely related derivative manufactured in a 0.35 micron process operating at a lower voltage. In the new process, bipolar transistors ceased to offer performance advantages and were removed at considerable design effort. Further derivatives of the same architecture migrated to 0.25 and 0.18 micron processes in which wire delay did not improve at the same rate as gate delay. Interconnect-dominated paths required further redesign to achieve good performance in the new processes. In contrast, the Pentium 4 was designed with process scaling in mind. Knowing that over the lifetime of the product, device performance would improve but wires would not, designers overengineered the interconnect-dominated paths for the original process so that the paths would not limit performance improvement as the process advanced [Deleganes02].

4.11 Historical Perspective

The incredible history of scaling can be seen in the advancement of the microprocessor. The Intel microprocessor line spans more than three decades. Table 4.19 summarizes the progression from the first 4-bit microprocessor, the 4004, through the Pentium 4, courtesy of the Intel Museum [Intel03]. Over the three decades, feature size has improved nearly one hundred-fold. Transistor budgets multiplied by more than 10,000. Even more remarkably, clock frequencies have also multiplied by almost as much. Even as the challenges have grown in the past decade, scaling has accelerated.

Table 4.19 History of Intel microprocessors over three decades

Processor	Year	Feature Size (μm)	Transistors	Frequency (MHz)	Word size	Package
4004	1971	10	2.3k	0.75	4	16-pin DIP
8008	1972	10	3.5k	0.5–0.8	8	18-pin DIP
8080	1974	6	6k	2	8	40-pin DIP
8086	1978	3	29k	5–10	16	40-pin DIP
80286	1982	1.5	134k	6–12	16	68-pin PGA
Intel386	1985	1.5–1.0	275k	16–25	32	100-pin PGA
Intel486	1989	1–0.6	1.2M	25–100	32	168-pin PGA
Pentium	1993	0.8–0.35	3.2–4.5M	60–300	32	296-pin PGA
Pentium Pro	1995	0.6–0.35	5.5M	166–200	32	387-pin MCM PGA
Pentium II	1997	0.35–0.25	7.5M	233–450	32	242-pin SECC
Pentium III	1999	0.25–0.18	9.5–28M	450–1000	32	330-pin SECC2
Pentium 4	2001	0.18–0.13	42–55M	1400–3200	32	478-pin PGA

Die photos of the microprocessors also illustrate the remarkable story of scaling. The 4004 [Faggin96] in Figure 4.71 was handcrafted to pack the transistors onto the tiny die. Observe the 4-bit datapaths and register files. Only a single layer of metal was available, so polysilicon jumpers were required when traces had to cross without touching. The masks were designed with colored pencils and were hand-cut from red plastic rubylith. Observe that diagonal lines were used routinely. The 16 I/O pads and bond wires are clearly visible. The processor was used in the Busicom calculator.

The 80286 [Childs84] in Figure 4.72 shows a far more regular appearance. It is partitioned into regular datapaths, random control logic, and several arrays. The arrays include the instruction decoder PLA and memory management hardware. At this scale, individual transistors are no longer visible.

The Intel486™ (originally 80486, but changed because a number cannot be trademarked) integrated an 8KB cache and floating point unit with a pipelined integer datapath, as shown in Figure 4.73. At this scale, individual gates are not visible. The center row is the 32-bit integer datapath. Above is the cache, divided into four 2KB subarrays. Observe that the cache involves a significant amount of logic beside the subarrays. Below are several blocks of synthesized control logic generated with automatic place and route tools. The "more advanced" tools no longer support diagonal interconnect. The wide datapaths in the upper right form the floating point unit.

The Pentium Processor™ in Figure 4.74 provides a superscalar integer execution unit and separate 8KB data and instruction caches. The 32-bit datapath and its associated control logic is again visible in the center of the chip, although at this scale, the individual bitslices of the datapath are difficult to resolve. The instruction cache in the upper left

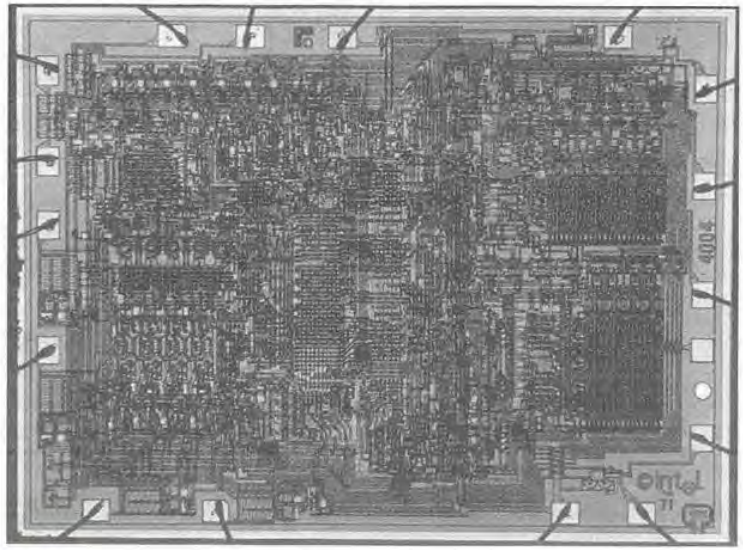


FIG 4.71 4004 microprocessor. Reprinted with permission of Intel Corporation.

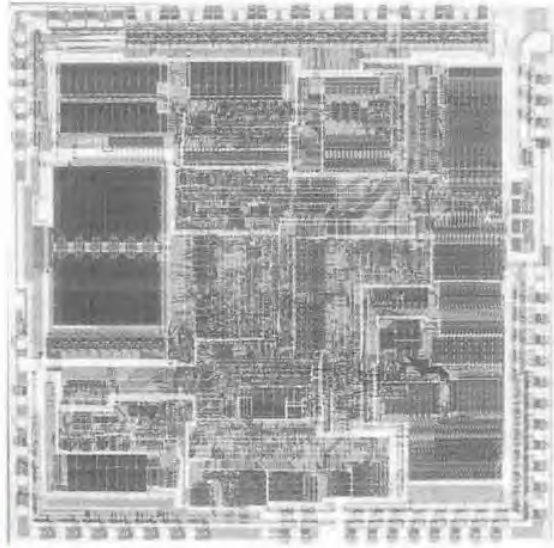


FIG 4.72 80286 microprocessor. Reprinted with permission of Intel Corporation.

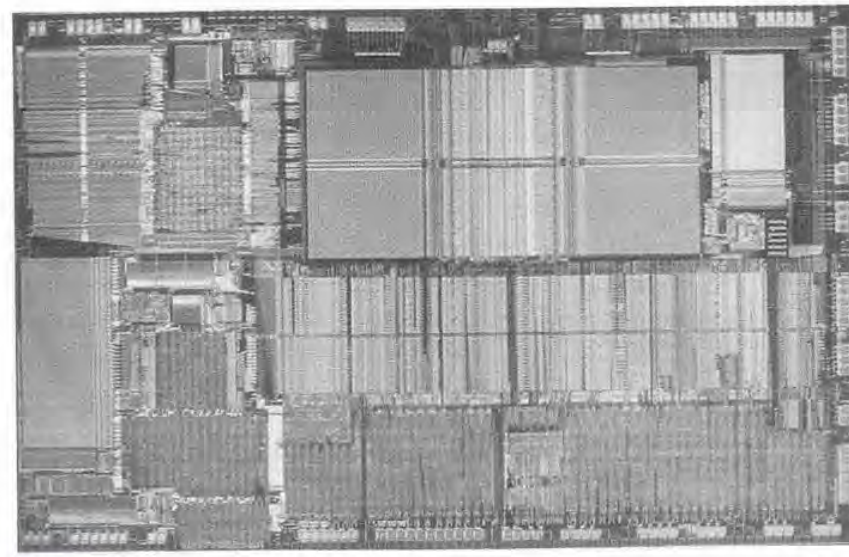


FIG 4.73 Intel486™ microprocessor. Reprinted with permission of Intel Corporation.

feeds the instruction fetch and decode units to its right. The data cache is in the lower left. The bus interface logic sits between the two caches. The pipelined floating point unit, home of the infamous FDIV bug [Price95], is in the lower right. This floorplan is important to minimize wire lengths between units that often communicate, such as the instruction cache and instruction fetch or the data cache and integer datapath. The integer datapath often forms the heart of a microprocessor, and other units surround the datapath to feed it the prodigious quantities of instructions and data that it consumes.

The Pentium™ III Processor, shown in Figure 4.75, offers out-of-order issue of up to three instructions per cycle. The entire left portion of the die is dedicated to 256–512 KB of level 2 cache to supplement the 32KB instruction and data caches. As processor performance outstrips memory system bandwidth, the portion of the die devoted to the cache hierarchy will continue to grow.

The Pentium™ 4 Processor is shown in Figure 4.76. The complexity of a VLSI system is clear from the enormous number of separate blocks that were each uniquely designed by a team of engineers. Indeed, at this scale even major functional units become difficult to resolve. The high operating frequency is achieved with a long pipeline using about 14 FO4 inverter delays per cycle. Remarkably, portions of the integer execution unit are "double-pumped" at twice the regular chip frequency.

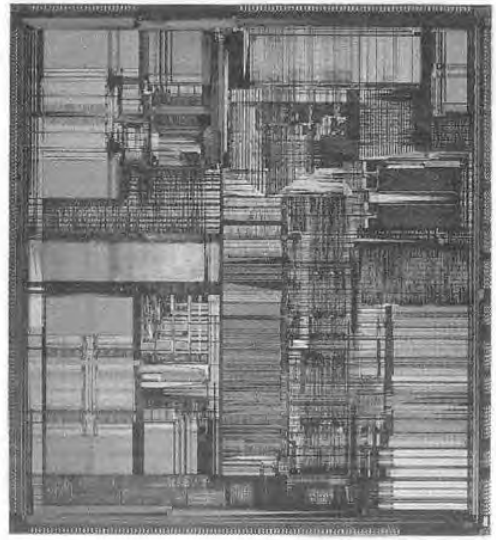


FIG 4.74 Pentium™ microprocessor. Reprinted with permission of Intel Corporation.

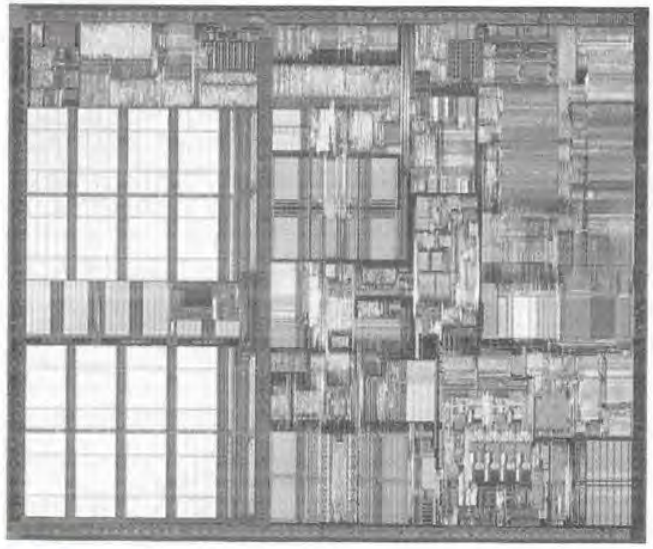


FIG 4.75 Pentium™ III microprocessor. Reprinted with permission of Intel Corporation.

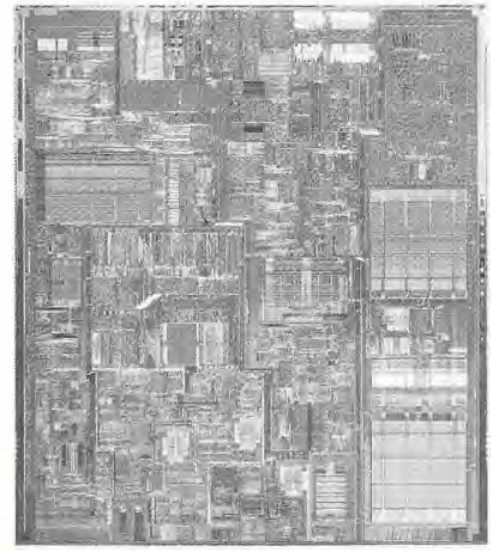


FIG 4.76 Pentium™ 4 microprocessor. Reprinted with permission of Intel Corporation.

Summary

The VLSI designer's challenge is to engineer a system that meets speed requirements while consuming little power or area, operating reliably, and taking little time to design. Circuit simulation is an important tool for calculating delay and will be discussed in depth in Chapter 5, but it takes too long to simulate every possible design; is prone to garbage-in, garbage-out mistakes; and doesn't give insight into why a circuit has a particular delay or how the circuit should be changed to improve delay. The designer must also have simple models to quickly estimate performance by hand and explain why some circuits are better than others.

Although transistors are complicated devices with nonlinear current-voltage and capacitance-voltage relationships, for the purpose of delay estimation in digital circuits, they can be approximated quite well as having constant capacitance and an effective resistance R when ON. Logic gates are thus modeled as RC networks. The Elmore delay model estimates the delay of the network as the sum of each capacitance times the resistance through which it must be charged or discharged. Therefore, the gate delay consists of a parasitic delay (accounting for the gate driving its own internal parasitic capacitance) plus an effort delay (accounting for the gate driving an external load). The effort delay depends on the electrical effort (the ratio of load capacitance to input capacitance, also

called *fanout*) and the logical effort (which characterizes the current driving capability of the gate relative to an inverter with equal input capacitance). Even in advanced fabrication processes, the delay vs. electrical effort curve fits a straight line very well. The method of Logical Effort builds on this linear delay model to help us quickly estimate the delay of entire paths based on the effort and parasitic delay of the path. We will use Logical Effort in subsequent chapters to explain what makes circuits fast.

The power consumption of a circuit has both dynamic and static components. The dynamic power comes from charging and discharging the load capacitances and depends on the frequency, voltage, capacitance, and activity factor. The static power comes from circuits that have an intentional path from V_{DD} to GND (like pseudo-nMOS) and from leakage. CMOS circuits have historically consumed relatively low power because complementary CMOS gates dissipate almost zero static power. However, leakage is increasing as feature size decreases, making static power consumption more important.

As feature size decreases, transistors get faster but wires do not. Interconnect delays are now very important. The delay is again estimated using the Elmore delay model based on the resistance and capacitance of the wire and its driver and load. The wire delay grows with the square of its length, so long wires are often broken into shorter segments driven by repeaters. Vast numbers of wires are required to connect all the transistors, so processes provide many layers of interconnect packed closely together. The capacitive coupling between these tightly packed wires can be a major source of noise in a system.

For reliable circuit operation, the designer must ensure that the circuit performs correctly across variations in the operating voltage and temperature, and must develop circuits that are robust for variations in transistor characteristics such as channel length or threshold voltage. Process corners are used to describe the worst-case combination of processing and environment for delay, power consumption, and functionality. The circuits must also be designed to operate correctly even as they age or are subject to cosmic rays and electrostatic discharge.

CMOS processes have been steadily improving for more than 20 years and will continue to do so for at least the next decade. A good designer not only should be familiar with the capabilities of current processes, but also should be able to predict the capabilities of future processes as feature sizes get progressively smaller. In modern constant-field scaling, gate delay improves with channel length. The number of transistors on a chip grows quadratically. The energy for each transistor to switch decreases with the cube of channel length, but the dynamic power density remains about the same because chips have more transistors switching at higher rates. Static power goes up as small transistors have exponentially more leakage. Interconnect capacitance per unit length remains constant, but resistance increases because the wires have a smaller cross-section. Local wires get shorter and have constant delay, while global wires have increasing delay. Architects planning future chips will take advantage of the larger number of faster transistors, but must also rethink existing architectures because of the changing ratio of wire-to-gate delay and the increasing leakage current.

Exercises

4.1 Sketch a 2-input NOR gate with transistor widths chosen to achieve effective rise and fall resistances equal to a unit inverter. Compute the rising and falling propagation delays of the NOR gate driving h identical NOR gates using the Elmore delay model. Assume that every source or drain has fully contacted diffusion when making your estimate of capacitance.

4.2 Sketch a stick diagram for the 2-input NOR. Repeat Exercise 4.1 with better capacitance estimates. In particular, if a diffusion node is shared between two parallel transistors, only budget its capacitance once. If a diffusion node is between two series transistors and requires no contacts, only budget half the capacitance because of the smaller diffusion area.

4.3 Find the rising and falling propagation delays of an AND-OR-INVERT gate using the Elmore delay model. Estimate the diffusion capacitance based on a stick diagram of the layout.

4.4 Find the worst-case Elmore parasitic delay of an n-input NOR gate.

4.5 Sketch a delay vs. electrical effort graph like that of Figure 4.8 for a 2-input NOR gate using the logical effort and parasitic delay estimated in Section 4.2.3. How does the slope of your graph compare to that of the 2-input NAND? How does the y-intercept compare?

4.6 Let a 4x inverter have transistors four times as wide as those of a unit inverter. If a unit inverter has three units of input capacitance and parasitic delay of p_{inv}, what is the input capacitance of a 4x inverter? What is the logical effort? What is the parasitic delay?

4.7 A three-stage logic path is designed so that the effort borne by each stage is 12, 6, and 9 delay units, respectively. Can this design be improved? Why? What is the best number of stages for this path? What changes do you recommend to the existing design?

4.8 Suppose a unit inverter with three units of input capacitance has unit drive.

a) What is the drive of a 4x inverter?

b) What is the drive of a 2-input NAND gate with 3 units of input capacitance?

4.9 Sketch a 4-input NAND gate with transistor widths chosen to achieve equal rise and fall resistance as a unit inverter. Show why the logical effort is 6/3.

4.10 Consider the two designs of a 2-input AND gate shown in Figure 4.77. Give an intuitive argument about which will be faster. Back up your argument with a calculation of the path effort, delay, and input capacitances x and y to achieve this delay.

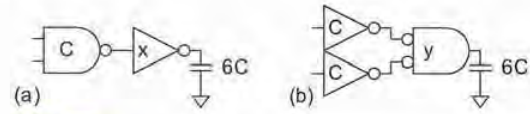


FIG 4.77 2-input AND gate

4.11 Consider four designs of a 6-input AND gate shown in Figure 4.78. Develop an expression for the delay of each path if the path electrical effort is H. What design is fastest for $H = 1$? For $H = 5$? For $H = 20$? Explain your conclusions intuitively.

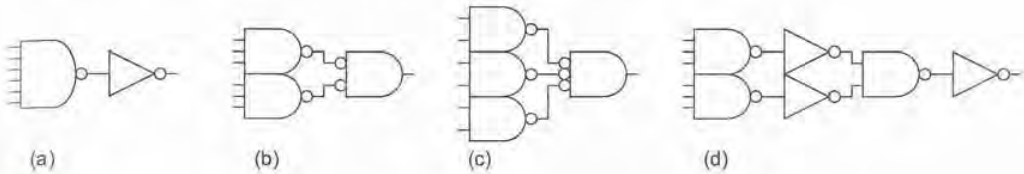


FIG 4.78 6-input AND gate

4.12 Repeat the decoder design example from Section 4.3.4 for a 32-word register file with 64-bit registers. Determine the fastest decoder design and estimate the delay of the decoder and the transistor widths to achieve this delay.

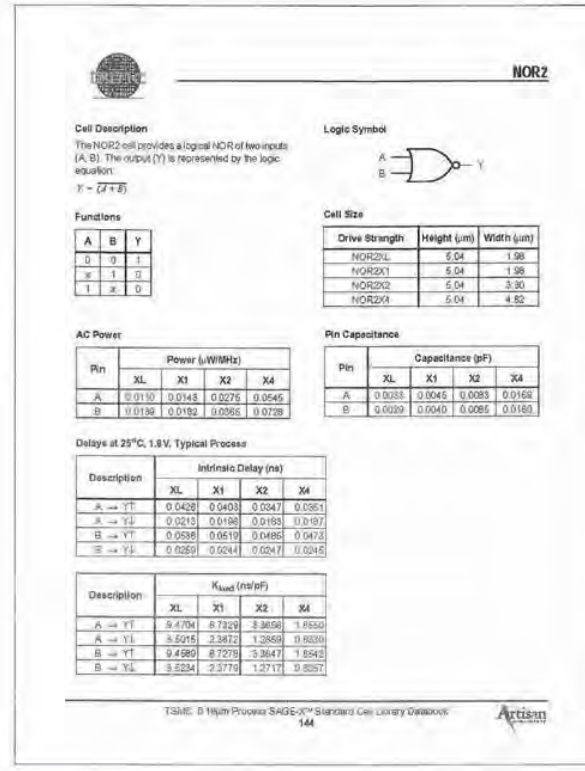
4.13 Design a circuit at the gate level to compute the following function:

```
if (a == b) y = a;
else y = 0;
```

Let a, b, and y be 16-bit busses. Assume the input and output capacitances are each 10 units. Your goal is to make the circuit as fast as possible. Estimate the delay in FO4 inverter delays using Logical Effort if the best gate sizes were used. What sizes do you need to use to achieve this delay?

4.14 Plot the average delay from input A of an FO3 NAND2 gate from the datasheet in Figure 4.25. Why is the delay larger for the XL drive strength than for the other drive strengths?

4.15 Figure 4.79 shows a datasheet for a 2-input NOR gate in the Artisan Components standard cell library for the TSMC 180 nm process. Find the average parasitic delay and logical effort of the X1 NOR gate A input. Use the value of τ from Section 4.3.7.



NOR2

Cell Description

The NOR2 cell provides a logical NOR of two inputs (A, B). The output (Y) is represented by the logic equation:

$Y = \overline{(A+B)}$

Logic Symbol

Functions

A	B	Y
0	0	1
x	1	0
1	x	0

Cell Size

Drive Strength	Height (µm)	Width (µm)
NOR2XL	5.04	1.98
NOR2X1	5.04	1.98
NOR2X2	5.04	3.30
NOR2X4	5.04	4.82

AC Power

Pin	Power (µW/MHz)			
	XL	X1	X2	X4
A	0.0110	0.0143	0.0276	0.0545
B	0.0139	0.0182	0.0365	0.0728

Pin Capacitance

Pin	Capacitance (pF)			
	XL	X1	X2	X4
A	0.0038	0.0045	0.0083	0.0169
B	0.0039	0.0040	0.0085	0.0160

Delays at 25°C, 1.8V, Typical Process

Description	Intrinsic Delay (ns)			
	XL	X1	X2	X4
A → YT	0.0428	0.0403	0.0347	0.0351
A → YL	0.0213	0.0196	0.0169	0.0197
B → YT	0.0536	0.0519	0.0486	0.0473
B → YL	0.0259	0.0244	0.0247	0.0245

Description	K_{load} (ns/pF)			
	XL	X1	X2	X4
A → YT	9.4704	4.7329	2.3656	1.1550
A → YL	5.5015	2.3872	1.2859	0.6830
B → YT	9.4689	4.7278	2.3647	1.1542
B → YL	5.5234	2.3779	1.2717	0.6857

TSMC 0.18µm Process SAGE-X™ Standard Cell Library Databook
144

Artisan

FIG 4.79 2-input NOR datasheet courtesy Artisan Components

4.16 Find the parasitic delay and logical effort of the X2 and X4 NOR gate *A* input. By what percentage do they differ from that of the X1 gate? What does this imply about our model that parasitic delay and logical effort depend only on gate type and not on transistor sizes?

4.17 What are the parasitic delay and logical effort of the X1 NOR gate *B* input? How and why do they differ from the *A* input?

4.18 Parasitic delay estimates in Section 4.2.4 are made assuming contacted diffusion on each transistor on the output node and ignoring internal diffusion. Would parasitic delay increase or decrease if you took into account that some parallel transistors on the output node share a single diffusion contact? If you counted internal diffusion capacitance between series transistors? If you counted wire capacitance within the cell?

4.19 Consider a process in which pMOS transistors have three times the effective resistance as nMOS transistors. A unit inverter with equal rising and falling delays in this process is shown in Figure 4.80. Calculate the logical efforts of a 2-input NAND gate and a 2-input NOR gate if they are designed with equal rising and falling delays.

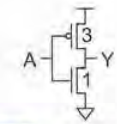


FIG 4.80 Unit inverter

4.20 Generalize Exercise 4.19 if the pMOS transistors have μ times the effective resistance of nMOS transistors. Find a general expression for the logical efforts of a k-input NAND gate and a k-input NOR gate. As μ increases, comment on the relative desirability of NANDs vs. NORs.

4.21 Some designers define a "gate delay" to be a fanout-of-3 2-input NAND gate rather than a fanout-of-4 inverter. Using Logical Effort, estimate the delay of a fanout-of-3 2-input NAND gate. Express your result both in τ and in FO4 inverter delays, assuming $p_{inv} = 1$.

4.22 Repeat Exercise 4.21 in a process with a lower ratio of diffusion to gate capacitance in which $p_{inv} = 0.75$. By what percentage does this change the NAND gate delay, as measured in FO4 inverter delays? What if $p_{inv} = 1.25$?

4.23 The 64-bit Naffziger adder [Naffziger96] has a delay of 930 ps in a fast 0.5-μm Hewlett-Packard process with an FO4 inverter delay of about 140 ps. Estimate its delay in a 70 nm process with an FO4 inverter delay of 20 ps.

4.24 An output pad contains a chain of successively larger inverters to drive the (relatively) enormous off-chip capacitance. If the first inverter in the chain has an input capacitance of 20 fF and the off-chip load is 10 pF, how many inverters should be used to drive the load with least delay? Estimate this delay, expressed in FO4 inverter delays.

4.25 The clock buffer in Figure 4.81 can present a maximum input capacitance of 100 fF. Both true and complementary outputs must drive loads of 300 pF. Compute the input capacitance of each inverter to minimize the worst-case delay from input to either output. What is this delay, in τ? Assume the inverter parasitic delay is 1.

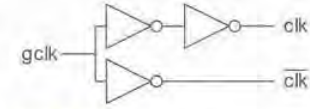


FIG 4.81 Clock buffer

4.26 The clock buffer from Exercise 4.25 is an example of a 1-2 fork. In general, if a 1–2 fork has a maximum input capacitance of C_1 and each of the two legs drives a load of C_2, what should the capacitance of each inverter be and how fast will the circuit operate? Express your answer in terms of p_{inv}.

4.27 A 180 nm standard cell process can have an average switching capacitance of 150 pF/mm^2. You are synthesizing a chip composed of random logic with an average activity factor of 0.1. Estimate the power consumption of your chip if it has an area of 70 mm^2 and runs at 450 MHz at $V_{DD} = 0.9$ V.

4.28 You are considering lowering V_{DD} to try to save power in a static CMOS gate. You will also scale V_t proportionally. Will dynamic power consumption go up or down? Will static power consumption go up or down?